IEEE MICRO

JUNE 1985



ALSO IN THIS ISSUE:

ROBOT-ARM CONTROL PROGRAM

8085A MULTIPROCESSOR SYSTEMS

SECOND-SOURCING CPUs

READER BALLOT: BEST ARTICLE OF 1984

IEEE COMPUTER SOCIETY

THE INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS, INC.

Reader Ballot

Best Article of 1984

Do you want to have an impact on the editorial flavor of *your* magazine? Vote for the article that, in your opinion, exemplified the best of *IEEE Micro* in 1984. Simply mark your preference on the Reader Interest Card at the back of the magazine.

February

101 L. Robert Morris, "Derek de Solla Price and the Antikythera Mechanism: An Appreciation," pp.15-21.
102 Derek de Solla Price, "A History of Calculating Machines," pp.22-52.
103 James M. Sibigtroth, "Motorola's MC68HC11: Definition and Design of a VLSI Microprocessor," pp. 54-65.
104 Ernst J. Schmitter and Peter Baues, "The Basic Fault-tolerant System," pp. 66-73.

April

105 M.F. Smith and B.E. Luff, "Automatic Assembler Source Translation from the Z80 to the MC6809," pp. 3-9.
106 Paul D. Stigall and Brian E. Lenharth, "A Microprocessor-controlled Message Display System," pp. 10-25.
107 Colin B. Hunter and Erin Farquhar, "Introduction to the NS16000 Architecture," pp. 26-47.
108 Daniel Tabak, "Dynamic Architecture and LSI Modular Computer Systems," pp. 48-66.

June

109 Borivoje Furht and Peter Lee, "An Efficient Software Driver for Am9511 Arithmetic Processor Implementation," pp. 7-19.
110 Walter S. Heath, "A System Executive for Real-Time Microcomputer Programs," pp. 20-32.
111 I/O Expansion Bus Subcommittee of IEEE Computer Society Microprocessor Standards Committee, "P959 I/O Expansion Bus Proposed Standard," pp. 33-54.
112 Peter U. Schulthess, "A Reduced High-Level-Language Instruction Set," pp. 55-67.

August

113 David B. Gustavson, "Computer Buses—A Tutorial," pp. 7-22.
114 R.V. Balakrishnan, "The Proposed IEEE 896 Futurebus—A Solution to the Bus Driving Problem," pp. 23-27.
115 D.M. Taub, "Arbitration and Control Acquisition in the Proposed IEEE 896 Futurebus," pp. 28-41.
116 Paul Borrill and John Theus, "An Advanced Communication Protocol for the Proposed IEEE 896 Futurebus," pp. 42-56.
117 Fletcher J. Buckley, "The IEEE Software Engineering Standards Process," pp. 57-62.
118 Don L. Jackson and Jack Cowan, "The Proposed IEEE 855 Microprocessor Operating Systems Interface Standard," pp. 63-71.
119 Tim Elsmore and William R. Shields, "The Proposed IEEE 1000 Microcomputer System Bus Standard," pp. 72-80.
120 Geoff Baldwin, "Towards an Assembly Language Standard," pp. 81- 85.
121 W.J. Cody et al., "A Proposed Radix- and Word-length-independent Standard for Floating-Point Arithmetic," pp. 86-100.
122 Doug MacGregor, Dave Mothersole, and Bill Moyer, "The Motorola MC68020," pp. 101-118.

October

123 Andrew Allison, "Microcomputer Peripherals—Status and Trends," pp. 9-17.
124 Roy J. Lahr, "The Non-Death of Paper," pp. 18-25.
125 P.D. Noakes and R. Aish, "A New Peripheral for Three-dimensional Computer Input," pp. 26-35.
126 Wolfgang Doster and Richard Oed, "Word Processing with On-Line Script Recognition," pp. 36-43.
127 Ernst Habekotte' and Stefan Cserveny, "A Smart Digital-Readout Circuit for a Capacitive Microtransducer," pp. 44-54.
128 James J. Farrell III, "The Advancing Technology of Motorola's Microprocessors and Microcomputers," pp. 55-63.

December

129 Barry W. Johnson, "Fault-tolerant Microprocessor-based Systems," pp. 6-21.
130 William F. McGill and Steven E. Smith, "Fault Tolerance in Continuous Process Control," pp. 22-33.
131 Richard Emmerson and Michael J. McGowan, "Fault Tolerance Achieved in VLSI," pp. 34-43.
132 C.S. Raghavendra, "Fault Tolerance in Regular Network Architectures," pp. 44-53.
133 Cedric V.W. Armstrong and Eli T. Fathi, "A Fault-tolerant Multimicroprocessor-Based Computer System for Space-Based Signal Processing," pp. 54-65.
134 Theodore J. Williams, "The Development of Reliability in Industrial Control Systems," pp. 66-80.

IEEE MICRO

Volume 5 Number 3 (ISSN 0272-1732) June 1985



On the cover
What materials does it take to efficiently package a high-performance microprocessor? Article on p.5.

Cover design: Jay Simpson and Larry Keiser

In the next issue: high-level languages; multiprocessing

Departments

4 Letters to the Editor
73 **MicroLaw**
Proprietary rights in cell libraries
79 **MicroReview**
The Juki 6300 daisywheel printer; short book reviews
80 **MicroStandards**
Eight long years . . . but success at last!
83 **MicroNews**
IEEE Micro authors win award; engineering workstations; Micromouse
86 **New Products**
93 **Product Summary**
96 **Calendar**
98 **MicroCourses**
99 **Access**
Recent books and articles on microcomputing
104 **Advertiser/Product Index; change-of-address form**
105 **Reader Service Card; Reader Interest Card**

Features

5 **Large-Scale Cost-Effective Packaging**
James J. Farrell III
VLSI circuits challenge traditional electronic packaging methods. Several new package types, however, efficiently handle high-I/O devices.

11 **An Evaluation of 8085-based Multiprocessing on a Timeshared Bus**
Brian E. Corrigan III and *Everett L. Johnson*
A GPSS simulation of a dual-processor system showed the effects of instruction type, bus access length, guard band length, clock frequency, and clock synchronization on its operation.

22 **A Microcomputer-based Control System for a Three-Joint Robot Arm**
Eric H. Stelzer and *Randy H. Moss*
This detailed investigation of the operating system and control program for a small model robot teaches lessons that the student can later apply in industry.

41 **Second-sourcing CPUs—Emulation, Ethics, and Electropolitics**
Chuck Hastings
Competitive games aimed at "locking in" customers to particular hardware or software will ultimately fail. Serving customers, not manipulating them, yields market success.

53 **Motorola's Silver Quill Program**
Jackie Marsh and *James J. Farrell III*
A well planned author incentive program can turn word-shy engineers into a company's best spokespersons.

Correspondence

58 **Binary-decision-based Programmable Controlle**
In certain applications, BD-based controllers can react more than conventional microprocessor-based controllers. *IEEE M* its original discussion of these devices with author- and reader-commentary.

IEEE Computer Society

Executive Committee

President: Martha Sloan
Department of Electrical Engineering
Michigan Technological University
Houghton, MI 49931
(906) 487-2845

Vice Presidents
Technical Activities (1st VP): Robert G. Stewart
Conferences and Tutorials (2nd VP): Roy L. Russo
Area Activities: Charles R. Vick
Educational Activities: J. Thomas Cain
Membership and Information: Russell E. Theisen
Publications: John D. Musa
Standards: Fletcher J. Buckley

Treasurer: Helen M. Wood
Secretary: Paul L. Borrill
Junior Past President: Oscar N. Garcia
IEEE Division Directors: Oscar N. Garcia,
Ronald G. Hoelzeman

Governing Board

Term Ending 1985	Term Ending 1986
James H. Aylor	Dennis R. Allison
Paul L. Borrill	Kenneth R. Anderson
Clyde R. Camp	P. Bruce Berra
Glen G. Langdon, Jr.	Fletcher J. Buckley
John F. Meyer	Judith L. Estrin
John D. Musa	Richard C. Jaeger
Harriett B. Rigas	Ming T. Liu
V. Thomas Rhyne	Hillel Ofek
Susan L. Rosenbaum	Edward W. Thomas
Herbert Weber	Joseph E. Urban

Publications Board

Dharma P. Agrawal	Glen G. Langdon, Jr.
James H. Aylor	Michael C. Mulder
Vic Basili	Theo Pavlidis
P. Bruce Berra	C. V. Ramamoorthy
Bill D. Carroll	T.R.N. Rao
James J. Farrell III	Susan L. Rosenbaum
Tse-yun Feng	Roy L. Russo
Dennis W. Fife	Norman Schneidewind
Paul L. Hazan	Bruce D. Shriver
Lansing Hatfield	James N. Snyder
Herbert Hecht	Murali Varanasi
Ronald G. Hoelzeman	Joseph E. Urban

John D. Musa, Vice President for Publications

Senior Staff

Executive Director: T. Michael Elliott
IEEE Computer Society
1730 Massachusetts Ave., NW
Washington, DC 20036-1903
(202) 371-0101

Editor and Publisher: True Seaborn
Director, Computer Society Press: Chip G. Stockton
Director, Conferences: William R. Habingreither
Director, Tutorials: Martez A. Camillilleri

Next Governing board meeting:
Chicago
July 17, 8:30 a.m. to 5 p.m.

The Institute of Electrical and Electronics Engineers, Inc.

President: Charles A. Eldon
President-Elect: Bruno O. Weinschel
Executive Vice President: Merlin G. Smith
Executive Director: Eric Herz

IEEE MICRO

Editor-in-Chief: James J. Farrell III, Motorola, Inc.*

Associate Editor-in-Chief:
Joe Hootman, University of North Dakota

Editorial Board:
George S. Carson, GSC Associates
David L. Hannum, AT&T Information Systems
Victor K. L. Huang, AT&T Information Systems
Barry W. Johnson, University of Virginia
Kenji Kani, Nippon Electric Company
Henriecus Koeman,
John Fluke Manufacturing Company
Richard Mateosian, National Semiconductor
L. Robert Morris,
Carleton University and DSPS Inc., Ottawa
Richard H. Stern
Robert G. Stewart, Stewart Research Enterprises

Advisors:
James H. Aylor, University of Virginia
J. Thomas Cain, University of Pittsburgh
Victor P. Nelson, Auburn University
Jean-Daniel Nicoud,
Swiss Federal Institute of Technology
Deene Ogden, Texas Instruments
Peter R. Rony, University of Delaware

Magazine Advisory Committee:
Dennis W. Fife (chairman), Dennis R. Allison,
Kenneth R. Anderson, James J. Farrell III (ed.-in-chief, *IEEE Micro*), Lansing Hatfield (ed.-in-chief, *IEEE CG&A*),
Ronald G. Hoelzeman, Stephen F. Lundstrom, Roy L. Russo
(ed.-in-chief, *IEEE Design & Test*),
Paul Schneck, Bruce D. Shriver
(ed.-in-chief, *IEEE Software*)

Editor and Publisher: True Seaborn
Managing Editor: Richard Landry
Contributing Editor: Joe Schallan
Art Director: Jay Simpson
Circulation Manager: Christina Champion
Advertising Director: Michael Koehler
Advertising Coordinator: Sandra J. Arteaga

Special thanks to Alex Torres and Miriam Wiegel for participating in the design of this issue.

*Submit six copies of all articles and special issue proposals to James J. Farrell III, 1505 Woodhill Drive, Round Rock, TX 78681; (512) 928-6572.

Circulation: *IEEE Micro* (ISSN 0272-1732) is published bimonthly by the IEEE Computer Society: IEEE Headquarters, 345 East 47th St., New York, NY 10017; IEEE Computer Society West Coast Office, 10662 Los Vaqueros Circle, Los Alamitos, CA 90720. Annual subscription: $12.00 in addition to IEEE Computer Society or any other IEEE society member dues. Nonmember prices: available on request. Single-copy prices: members $7.50; nonmembers $15.00. This journal is also available in microfiche form.

Undelivered copies: Send to 10662 Los Vaqueros Circle, Los Alamitos, CA 90720.

Postmaster: Send address changes to *IEEE Micro*, IEEE Service Center, 445 Hoes Lane, Piscataway, NJ 08854. Second class postage is paid at New York, NY, and at additional mailing offices.

Copyright and reprint permissions: Abstracting is permitted with credit to the source. Libraries are permitted to photocopy beyond the limits of US Copyright Law for private use of patrons: those post-1977 articles that carry a code at the bottom of the first page, provided the per-copy fee indicated in the code is paid through the Copyright Clearance Center, 29 Congress St., Salem, MA 01970. Instructors are permitted to photocopy isolated articles for noncommercial classroom use without fee. For other copying, reprint, or republication permission, write to Editor, *IEEE Micro*, 10662 Los Vaqueros Circle, Los Alamitos, CA 90720. All rights reserved. Copyright © 1985 by the Institute of Electrical and Electronics Engineers, Inc.

Editoral: Unless otherwise stated, bylined articles, as well as products and services offered in New Products, the Product Summary, MicroReview, MicroNews, and Access, reflect the author's opinion; inclusion in this publication does not necessarily constitute endorsement by the IEEE or the IEEE Computer Society.

BPA

From the Editor-in-Chief



On behalf of the entire staff of *IEEE Micro,* I would like to thank Pat Buffett, John Burkitt, Peter Rony, Deene Ogden, Tom Cain, Jim Aylor, and Jean-Daniel Nicoud, who have all completed their editorial terms. Their efforts and direction were essential to making *IEEE Micro* the high-quality, definitive technology magazine that it is. Rony, Ogden, Cain, Aylor, and Nicoud have consented to remain in a noneditorial (they will not review articles), advisory capacity for one year.

Please welcome three new members of the editorial board: Joe Hootman of the University of North Dakota, our new Associate Editor-in-Chief; and our new Associate Editors Barry Johnson of the University of Virginia and Victor K.L. Huang of AT&T Information Systems. We will formally introduce them in the August issue.

While the Editorial Board is responsible for columns, technical direction, and for reviewing contributed technical articles, it is the Managing Editor who puts the magazine together. He copyedits the submitted manuscripts and columns (including this one), so that they appear in the Queen's English and make sense when read. He is responsible for all aspects of producing the magazine, from defining and selecting the cover to making sure it is mailed to you on time.

We are fortunate to have Richard Landry as our Managing Editor, on site, in Los Alamitos. Richard's biography and photograph appear on p. 85.

Last April, when I visited Los Alamitos, Richard, Joe Schallan (who remains as a Contributing Editor), and I discussed some changes and additions to *IEEE Micro.* Since then Richard has been busy.

Please notice the three new features of *IEEE Micro* which we hope will increase the value of the magazine to you and on which we would like you to comment. First, on the inside front cover of the magazine you will find a ballot for the best *IEEE Micro* article of 1984. We believe that our readers are the best judges of the material we present in *IEEE Micro,* so we are calling for your help to recognize the author or authors who, in the opinion of the readership, contributed most to the magazine last year. To vote for the best article of 1984, simply fill out the Reader Interest Card at the back of the magazine (more about the Reader Interest Card below).

Also, in this issue *IEEE Micro* bids goodbye to its old format, which has served us well for over four years. The new format, which we hope you will enjoy, stresses greater ease of reading. Our graphics are now intended to convey as much information as possible at a glance; and the revised table of contents and convenient headings at the top of each page make it a simple task to browse through the entire magazine or to find a favorite department.

Finally, because your interest in and enjoyment of *IEEE Micro* is most important to us, we have begun with this issue to include a special Reader Interest Card at the back of the magazine. This card, which is separate from the Reader Service Card that you fill out to learn more about new product offerings, comes directly to the *IEEE Micro* Editorial Board and staff. Filling it out is one very potent way of making sure that the editorial focus of *IEEE Micro* is to your liking. Every article and department in the magazine will now carry a Reader Interest Survey. Just circle the appropriate number on the Reader Interest Card to let us know whether your interest in that portion of the magazine was high, medium, or low. And, while you are filling out the card, take a few moments to jot some comments on any subject matter you would like us to cover or any ideas you might have on how we can better serve you.

Best wishes,

Jim Farrell

Jim Farrell

To the Editor

More on big-endian vs. little-endian byte ordering

To the Editor:

I prefer a different summary of my big-endian versus little-endian views than that offered by Tim Paterson (Letters, *IEEE Micro,* February 1984, page 11).

Let us use the terminology "high-first" and "low-first," instead of "big-endian" and "little-endian," to reduce the emotional content and clarify the meaning. "High-first" means that the most significant (high) byte of an integer is stored at the lowest (first) address.

The arguments favoring high-first byte ordering and those favoring low-first byte ordering cancel each other, when properly paired, except for one: Character strings are addressed most-significant-byte-first in both schemes, but multibyte integers or addresses are addressed least-significant-byte-first in the low-first one.

This low-first asymmetry is the cause of most of the inconveniences which plague users of that scheme. It also causes most of the difficulty in interfacing the two schemes, because it creates a need for context-dependent byte swapping. Since context information (the "meaning" of the bytes) is not usually available to hardware or to general software utilities, transforming data from one scheme to the other is intrinsically difficult and requires ad hoc algorithms for every application.

The inconsistent numbering of bits and bytes often used in today's high-first microprocessors is inelegant; however, in practice the bit numbering is orders of magnitude less important than the byte-addressing problem, which causes significant wasted time for both humans and computers. This is because the significance of bits within bytes is the same in all contexts—only the bit number labels differ between the two systems, and in practice they are rarely used in a way that affects data portability.

New designs ought to use the high-first scheme for both byte addressing and bit numbering. However, it is not a serious problem if they use the low-first scheme for bit numbering, as the MC68000 and Z8000 do. By adopting the main virtue of the low-first scheme, these practical hybrids have the best of both, paying only a minor penalty for their slight inconsistency.

Of course, we must forgive those designers who, because of compatibility considerations, are constrained to the optimizations and errors of the past. That explains the 8086, which is an enhanced 8080, which is an enhanced 8008, whose address arithmetic was simplified by storing least significant bytes first in memory. But how sad that the otherwise elegant PDP-11 and the NS16000 adopted the internally inconsistent scheme.

David B. Gustavson
Stanford Linear Accelerator Center
PO Box 4349, Bin 88
Stanford, CA 94305

Semaphore strategy for Z80

To the Editor:

In his article, "A System Executive for Real-Time Microcomputer Programs" (*IEEE Micro*, June 1984, pp. 20-32), Walter S. Heath implies that a semaphore cannot be directly supported by the Z80 microprocessor because of the lack of a test-and-set instruction. The solution proposed is to simulate the test and set by first disabling the interrupts. This is a common criticism of many eight-bit micros. However, in many cases, including the Z80, there exists instead a set-and-test mechanism which allows direct implementation of a semaphore without circuitous programming.

This mechanism is the shift in memory. My own preference is the shift right logical in memory (Z80 mnemonic SRL). Utilizing this instruction, one can support a semaphore through the following procedure:

1. Initialize the memory semaphore by setting only its least significant bit (LSB).
2. To set the semaphore perform the SRL. This shifts the LSB into the C flag in a single indivisible operation.
3. Test the C flag. If set, the semaphore is open. If clear, the semaphore is locked.
4. If open, proceed to use the now-protected resource. When finished, unlock the resource by writing a one to the semaphore location.

Since operation two is indivisible, it effectively locks out any subsequent or interrupting task seeking access to a protected resource.

Wolfram Lunscher
Research & Development
Develcon Electronics Ltd.

Reader Interest Survey

Indicate your interest in this department by circling the appropriate number on the Reader Interest Card.

High 176 Medium 177 Low 178

Large-Scale Cost-Effective Packaging

James J. Farrell III
Motorola, Inc.

VLSI circuits challenge traditional electronic packaging methods. Several new package types, however, efficiently handle high-I/O devices.

The packaging of integrated circuits has challenged the electronics industry ever since their invention by Jack Kilby and Robert Noyce. Kilby and Noyce's concept dramatically reduced the amount of physical area consumed by a circuit. Over the past 25 years, the overwhelming success of that concept has caused integration at ever higher levels. The earliest IC solutions permitted hundreds of transistors to be placed in a small area; today's designs require hundreds of thousands of transistors to be placed in the same area. As a result, getting the connections from the printed-circuit board to the highly integrated die in a space-effective manner has become the chief problem in IC packaging.

Packaging technology for today's myriad applications has not produced a single, all-purpose "panacea" package. Here, I will examine some of the more popular solutions and discuss some trends in the field.

The name of the game in packaging is to allow the multiple-I/O integrated circuit to be interfaced to the rest of the system, usually via a printed-circuit board. Figure 1 shows the range of pin densities encountered with three package types.

DIP

By far, the most popular package for integrated circuits is the DIP—the dual inline package—in all its variations. Figure 2 shows a typical DIP. Well over 90 percent of all integrated circuits shipped in 1983 were in DIPs, and this percentage dropped only slightly in 1984 because of the increasing popularity of other package types. It is also estimated that over 1.5 *billion* integrated circuits were shipped in 1984 in the US alone, due primarily to the general economic recovery. The pinout on DIPs ranges from 6 to 64, with the overwhelming majority of devices being in the 14-to-40-pin range.

The DIP has proved to be an extremely utilitarian package for most applications. Its short, stiff leads on 0.1-inch centers allow reasonably easy insertion both by hand and by automatic insertion devices (except for the largest and smallest types). It is generally available in three materials, plas
most popular (85 percent) and
cheapest. The plastic package
usually cannot qualify for ap
hostile environments in whic

0272-1732/85/0600-0005$01.00 © 1985 IEEE

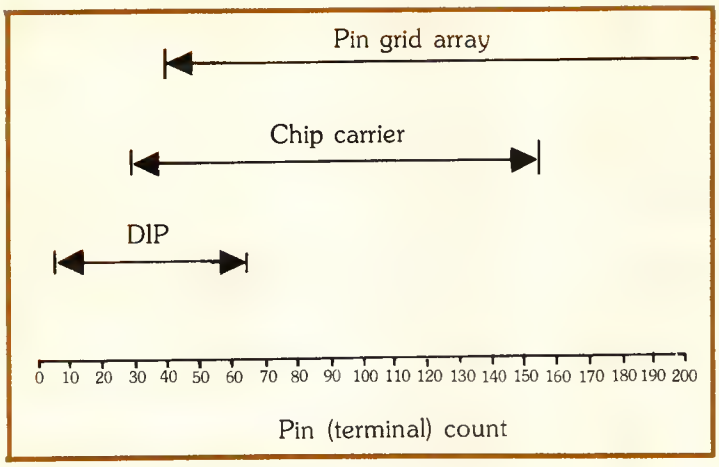
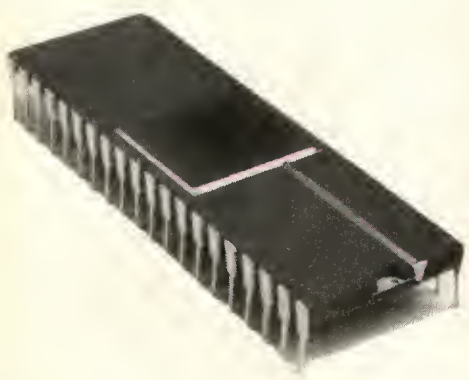


Figure 1. IC package pin densities.

cold, acids, salt water, or high humidity are present. Nor can it qualify for military applications. It can be socketed or soldered directly onto the printed-circuit board. The cost of the device as well as its mean time between failure are usually the factors in any decision to use it. The second most popular type (14 percent) is the cerdip, a cost-compromise package, that is actually a ceramic "sandwich." In this package type, the lead frame of the plastic DIP is nested between two ceramic slabs which are cemented together with an epoxy-like material. This package provides many of the environment-tolerant features missing in the plastic DIP at a moderate cost premium. The fully ceramic DIP represents a relatively small part of the market (1 percent), but an important one. The ceramic DIP, frequently referred to as "side-brazed," is an expensive package. It frequently utilizes gold plating on both its leads and lid, and can actually be more costly than the die it houses. However, this package must be used in most military and other harsh environments in order to comply with specification demands.

The range of DIP sizes and pinouts has distinct limits. While there are some DIP-like packages for two-leaded capacitors, the true DIP has at least four or six leads. There is some activity at the 64-pin level, caused primarily by the Motorola MC68000 microprocessor, but, as stated before, the overwhelming majority of the packages are in the 14-to-40-pin range. The 64-pin DIP revealed some difficult mechanical problems. A 64-pin DIP with 0.1-inch pin centers consumes nearly three square inches of printed-circuit board space. In addition, the part is difficult to handle, test, stock, ship, and insert. On the other hand, there are some mechanical advantages that still make the DIP desirable even in this high pin count. It can be easily designed into single-sided printed-circuit boards using



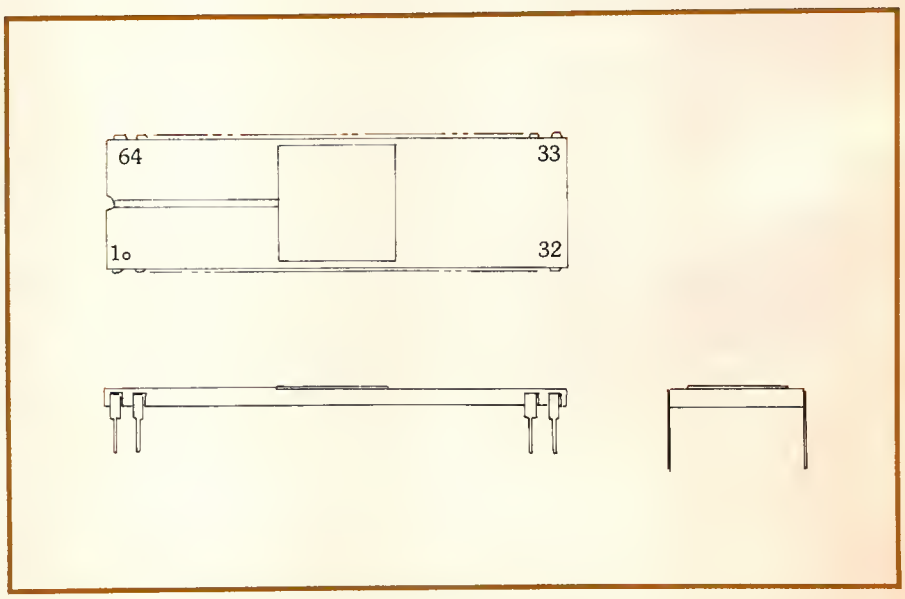


Figure 2. Dual inline package.

standard 0.1-inch copper design rules. This is very important to low-cost applications that cannot afford double-sided or multilayer printed-circuit boards, 0.05-inch layout rules, surface mounting techniques, or expensive sockets and connectors.

Flatpak

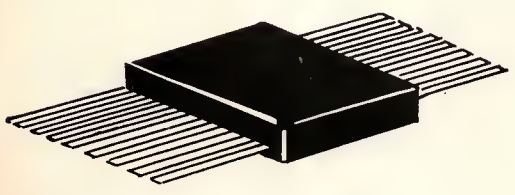
The flatpak (Figure 3), while second in popularity only to the DIP types, holds less than 0.5 percent of the IC package market. Furthermore, this market share is eroding and the total number of devices packaged in this way is expected to decline. It is projected that in the late 1980's, less than 0.1 percent of the ICs shipped will be housed in this package. The flatpak is small and flat and has fairly flimsy leads in the same plane as the package. It is somewhat hard to handle and test and is usually surface-mounted.

Chip carriers

The ceramic chip carrier and the plastic leaded chip carrier, or PLCC, presently comprise less than one percent of the IC package market. However, this percentage is expected to surge dramatically in the next five years to over 8 percent. Most chip carriers are used in computer memory, telecommunications, and military applications in ceramic. The ceramic chip carrier's small size and hermaticity, and its other mechanical advantages, have made it very desirable for these applications. The major disadvantage of this ceramic leadless device is its inability to be soldered directly to materials such as the standard fiberglass used in printed-circuit boards. It must first be soldered to an alumina material that has a coefficient of expansion similar to its own, or, alternatively, a connector (socket) must be used.

For the most part, ceramic chip carriers and PLCCs have termination spacings of 0.05 inch, although 0.04-inch spacing can frequently be found on smaller packages such as those used in pocket pagers and in other applications in which space is at a premium. The 0.05-inch design rules for printed-circuit boards that use this package can be overcome by using a connector whose footprint is oriented to 0.1-inch design rules.

The PLCC overcomes the surface mounting problems of the ceramic package by having compliant or movable leads. The ability of the leads to move and compensate for dissimilar coefficients of expansion makes this part solderable in many applications. The PLCC is



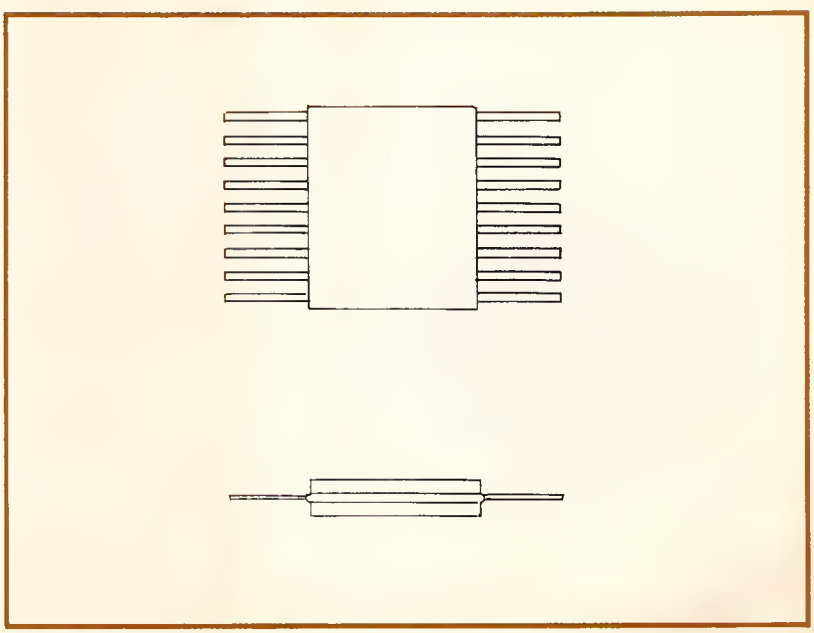
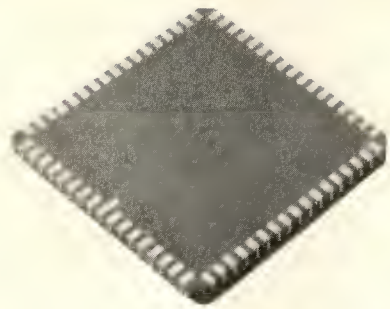


Figure 3. Flatpak.



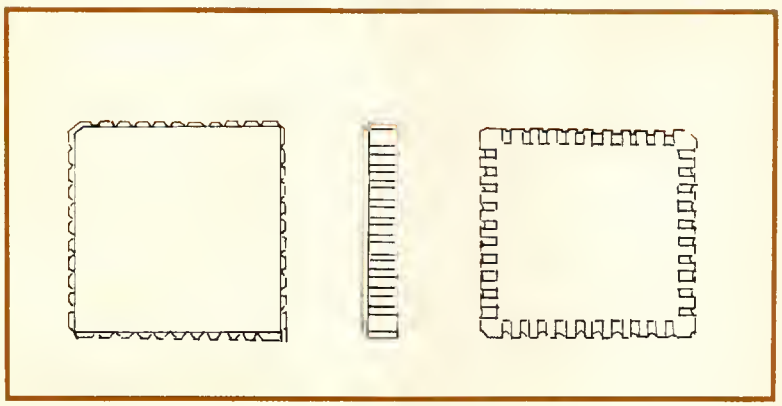


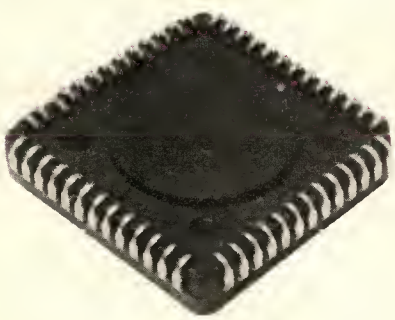
Figure 4. Chip carrier.

beginning to look very attractive to designers in applications requiring large amounts of memory or ICs with many inputs/outputs, and in applications having printed-circuit board real-estate restrictions.

Figure 4 shows the ceramic chip carrier, and Figure 5 the PLCC.

Pin grid arrays

Pin grid arrays, or PGAs, are relatively new on the packaging scene. Presently, they represent an insignificant portion of the IC packaging market (less than 0.1 percent). However, PGAs are coming on strong in several areas that are very important. The overwhelming majority of PGAs are ceramic and thus have all the desirable mechanical characteristics of ceramic DIPs or chip carriers. Furthermore, for a given number of inputs/outputs, they are about the same size as a chip carrier. They are frequently referred to as chip carriers with leads. The leads are usually on 0.1-inch centers, making layout of printed-circuit boards easier. Another advantage of PGAs is



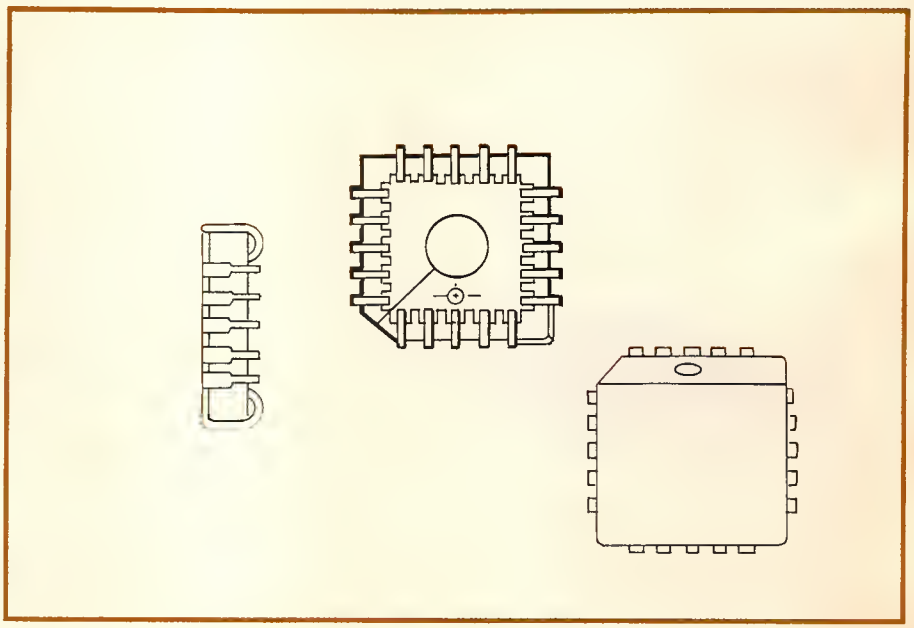


Figure 5. Plastic leaded chip carrier.

their ability to handle over 200 pins. PGAs accommodate these large numbers of pins in two rows around the periphery or in a solid matrix of pins. There is one danger here, however. If the PGA pin count is high enough—typically over 68 pins—a multilayer printed-circuit board may be required.

PGAs can be socketed or soldered directly onto printed-circuit boards. However, the cost savings derived from soldering directly onto boards does not overcome the higher expense of the PGA package as compared to that of

Table 1.
Package characteristics.

Features	JEDEC leadless chip carriers A	B[1]	C[1]	DIP Ceramic	Plastic	Plastic leaded chip carrier	Pin-grid-array ceramic
Sockets/connectors available	Yes	Yes	No	Yes	Yes	Yes	Yes
Direct solder-on	No	No	Yes[2]	Yes	Yes	Yes	Yes
Pin/terminal count, min./max.	14/156	14/156	14/156	6/64	6/64	16/156	40/225
Pin spacing (mils)	50	50	50	100	100	50	100
Lead resistance (R max. in ohms)	0.2	0.25	0.5	0.2	0.1	0.1	0.25
Lead capacitance (C max. in. pF)[3]	2	2	2	7	4	2	2
Package cost[4,7]	3	4[5]	5[5]	2	7	6	1
Requires connector/socket	Yes	Yes	No	No	No	No	No
Minimum mounting cost[4,7]	1	2	5	4	7	6	3
Minimum PC board area (in.[2])[6,7]	1.25-1.35	1.25-1.35	1.0	2.9	2.9	1.0	1.0
Maximum height when mounted (in.)[6,7]	0.25-0.45	0.25-0.45	0.12	STD	STD	0.15	0.2

Notes:
1. Both B and C have the same footprint and terminal numbering.
2. On alumina substrate.
3. The lead capacitance varies little between the A, B, and C LCC packages.
4. This rating is relative, with 1 being the most expensive and 7 being the least expensive.
5. Equal costs.
6. This is the range of sizes for connectors that we know of.
7. For these examples an MC68000 (64 I/O) was used.

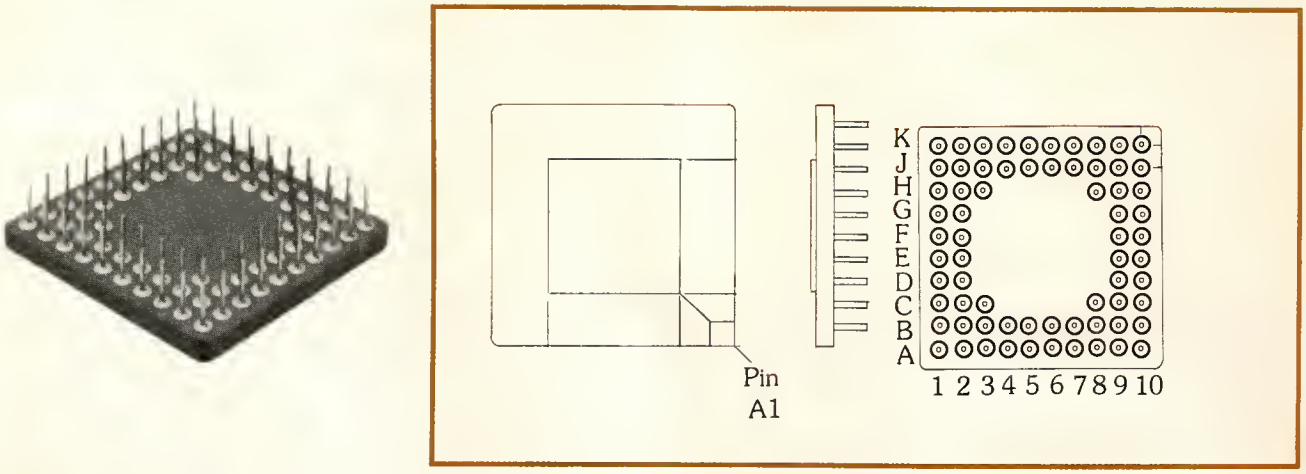


Figure 6. Pin grid array.

the chip carrier and DIP. Table 1 gives the attributes of pin grid arrays compared to those of PLCCs, dual inline packages, and leadless chip carriers. Figure 6 shows a typical PGA.

Although electronic packages will continue to evolve, the traditional plastic or ceramic DIP can be expected to retain the largest share of the market for some years to come. The flatpak's future is limited, but chip carriers and PLCCs should see much greater use by the end of the decade. The pin grid array blends the mechanical advantages of the DIP with the ability to support high pin counts—it may be particularly attractive for the high I/O VLSI circuits of the future.

James J. Farrell III is manager of technical communications for Motorola's Semiconductor Products Sector in Austin, Texas. He graduated from the United States Armed Forces Institute (Tokyo) and also attended the New Jersey Institute of Technology. Prior to his present position, Jim held the positions of manager of technical information for microprocessors and application engineer for microprocessors. Before coming to Motorola in 1977, he spent 10 years with Electronic Associates, Inc., engaged in analog/hybrid computer development. Prior to joining EAI, Jim worked on hybrid integrated circuits and power transistor applications for Bendix Semiconductor in Holmdel, New Jersey.

Jim has published or presented over 50 articles and papers in the United States and abroad, in *Electronics, Electronic Design, EDN, Digital Design, Electronic Products, New Electronics* (China), *Elektroniker* (Switzerland) and *Australian EE*, and at Electro, Wescon, Midcon, IMAC (Japan), and Nikkei (Japan), among others. He is the Editor-in-Chief of *IEEE Micro*. Jim's hobby is geology.

The author's address is Technical Communications, Motorola, Inc., 3501 Ed Bluestein Blvd., Austin, TX 78721.

Reader Interest Survey

Indicate your interest in this article by circling the appropriate number on the Reader Interest Card.

High 150 Medium 151 Low 152

An Evaluation of 8085-based Multiprocessing on a Timeshared Bus

Brian E. Corrigan III
Boeing Military Airplane Company

Everett L. Johnson
Wichita State University

A GPSS simulation of a dual-processor system showed the effects of instruction type, bus access length, guard band length, clock frequency, and clock synchronization on its operation.

Design for maximum efficiency and utilization of the more expensive elements has been a goal in the design of computer systems. The cost of system building blocks relative to one another has changed significantly in recent years. Today, peripherals, not processors, represent the most costly design elements. Advances in semiconductor technology have reduced the cost of CPUs, ROMs, RAMs, and I/O modules. All of these elements cost less than the peripherals in a system.

To achieve maximum efficiency, the designer must keep the expensive peripherals as busy as possible. Fast peripherals that interface directly to local storage through DMA can be used to the limit of performance. On the other hand, many applications require that the transfer rate of data to/from the peripheral be limited by the speed of the processor. Microprocessor speeds have not kept pace with memory speeds. Although they may require fast-access memories, none of the currently popular microprocessors can effectively utilize the data transfer rates possible with members of today's ROM and RAM families. By using two or more processors in a system to share high-speed RAM and ROM, however, the designer can improve performance and realize a greater utilization of system elements.

In addition to supporting faster data transfer rates, an architecture consisting of multiple processors usually attempts to exploit the parallelism inherent in the jobs that are to be executed. Because two or more processors are operating simultaneously, a much higher throughput is possible. The computation in a job is divided into subtasks to be executed by the different processors. Such subtasks include matrix operations, computation of statistical moments of arrivals of data samples, and presentation of output data to devices such as printers and plotters.[1]

Multiple processors and memories can be interconnected in two ways—either as a *multiprocessor* system or as a *multicomputer* system. In a multicomputer system, the processors, with attached local memory units, are interconnected only by message-passing buses. This arrangement is also termed a loosely-coupled system. Memories are shared only in the sense that a processor may transmit

0272-1732/85/0600-0011$01.00 © 1985 IEEE

messages to, and receive them from, other processor-memory pairs.[2] In a multiprocessor system, the processors are connected to a single, shared main memory through an interconnection network. This arrangement is also known as a tightly coupled system.[2,3] There are several types of processor/memory interconnection switches and networks. These include full crossbar switches, which are the most expensive, delta networks,[3] and timeshared buses,[4] which are the least expensive.

Timesharing of the multiprocessor bus

Here, we are concerned with multiprocessing attained through the use of a timeshared bus. Most microprocessors are able to support shared-bus multiprocessor systems. Intel's 808X family uses HOLD and HLDA, or RQ and GT, signals to provide bus control switching. Zilog's Z8X family has the signals BUSREQ and BUSACK. Use of these signals for shared-bus operation does not in itself provide the desired multiprocessing capability, since at a given time only one processor is active on the bus. However, the timing of the machine cycles makes it possible to use bus control and arbitration logic to mesh the bus accesses of two or more processors. They become transparent to each other with little degradation in performance. This method has been applied to the Intel 8086 by Yue and Halverson.[4] Figure 1 shows the bus cycle timing diagram for the Intel 8086. Each bus access or transaction requires at least four clock cycles. Clock cycle T1 is used to issue and latch the memory address to be accessed. Data written out and control signals become stable during clock cycle T2 and remain so until clock cycle T4. Input data are sampled near the trailing edge of either the T3 clock cycle or the wait (TW) clock cycle that is inserted between T3 and T4. The address latch enable signal that occurs at the beginning of clock cycle T1 begins every bus transaction. Read transactions end 10 ns after the trailing edge of the T3 clock cycle. A bus read can occur without accessing the bus during the T2 or T4 clock cycle. Write transactions start at the beginning of clock cycle T2 and end at the end of clock cycle T3. Bus writes can end during clock cycle T2, since all data needed are available at that time. The bus is active during T2 for bus writes, and during T3 for bus reads. On the bus, clock cycles T2 and T3 are active, and clock cycles T1 and T4 inactive, as far as the memory connected to the bus is concerned. This idle-bus time can be used by another processor in a bus/memory timesharing mode.

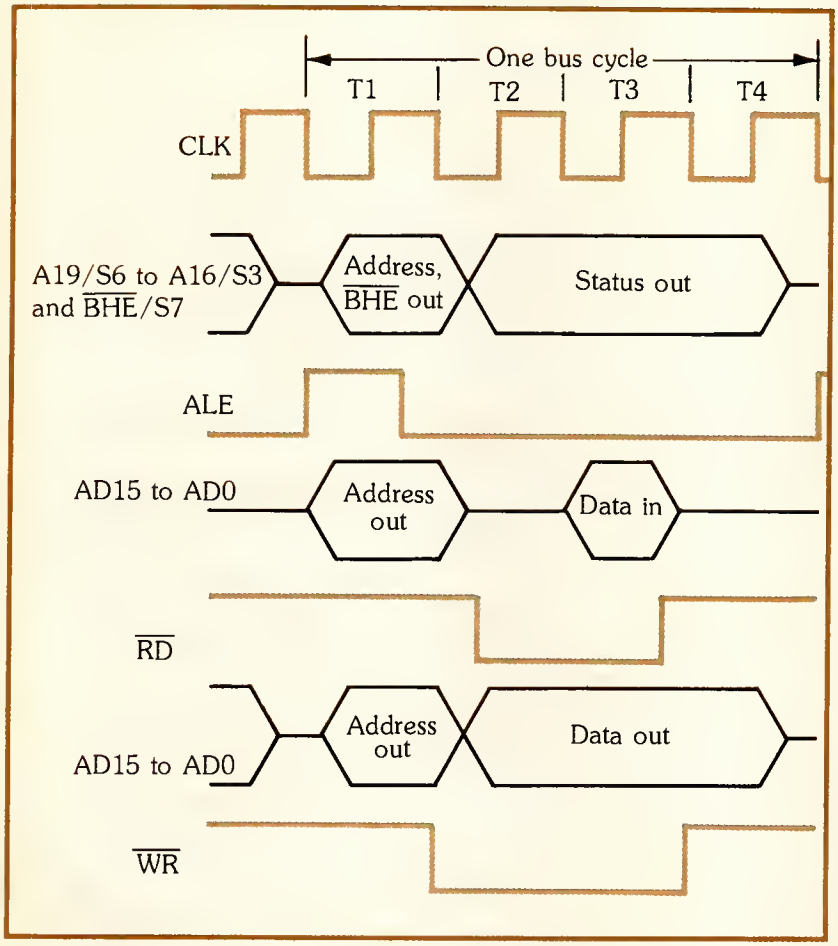


Figure 1. Timing diagram for the 8086 microprocessor.

Our main objective here is to apply this timesharing method of multiprocessing to an 8085 system like the one shown in Figure 2. The machine cycle timing diagrams for the 8085A are shown in Figure 3. The read bus access begins at the beginning of the T2 clock cycle and ends in the middle of the T3 clock cycle. The write bus access begins at the begin-

ning of the T2 clock cycle and ends in the middle of the T3 clock cycle. In a timesharing system, if the speed of the memory permits it, the bus access can be reduced from 1.5 clock cycles to 1 clock cycle and can end at the beginning of the T3 clock cycle. This requires a register (such as the interim register in Figure 2) to store the data to be read by the processor, since the 8085A does not load the data read into its registers until it encounters the clock edge in the middle of the T3 clock cycle. However, if the bus access time is kept

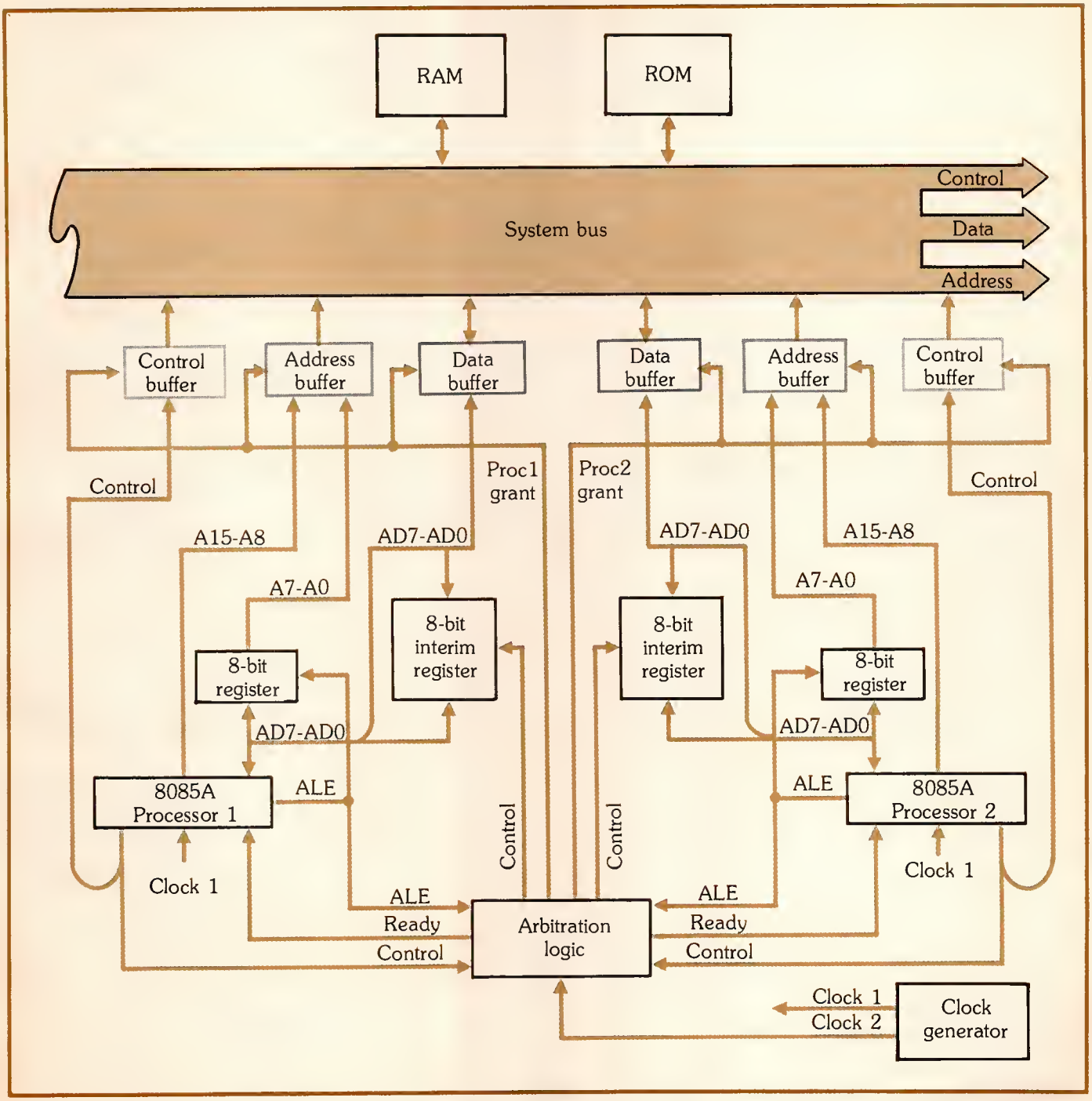


Figure 2. An 8085A multiprocessor system.

at 1.5 clock cycles, additional wait states have to be inserted into the machine cycles of the processors. We will examine both the 1-cycle case and the 1.5-cycle case.

The ideal situation would be if a bus access by one processor could occur immediately after the completion of a bus access by another processor. In reality, a guard band (a short time interval) must be inserted between each bus access to allow the address, data, and control lines to settle. Here, we consider systems using no guard band (to establish the ideal improvement), a guard band of one-half of a clock cycle, and a guard band of one complete clock cycle. Figure 4 shows examples of processor timing, under each guard-band condition, for two processors timesharing the same bus, where the bus access time is one clock cycle. (The interim register holds the data read from the bus until the data are requested by the processor.) Figure 5 shows examples of processor timing for two pro-

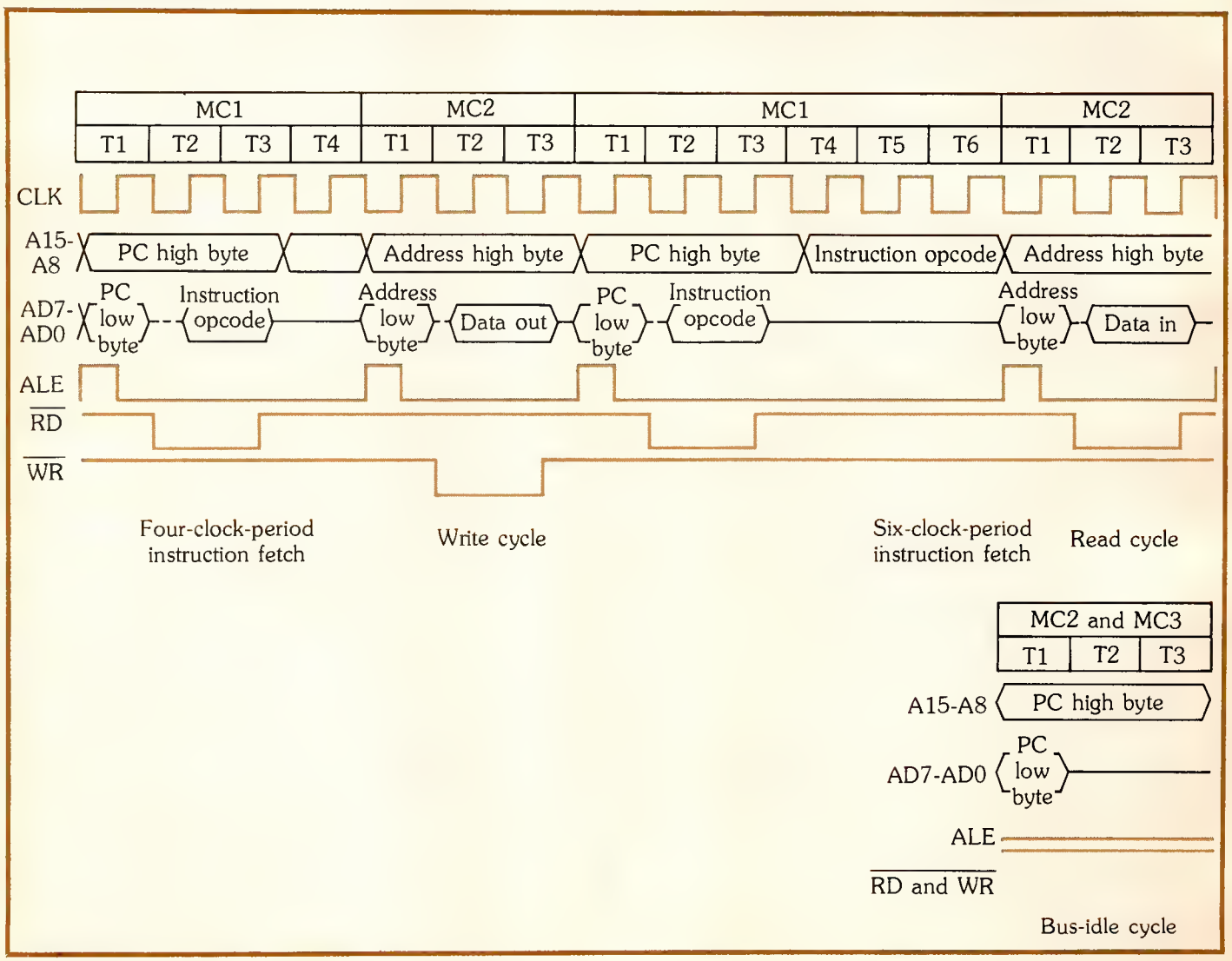


Figure 3. Machine cycle timing for the 8085A.

cessors timesharing the same bus, where the bus access time is 1.5 clock cycles. Again, performance under each guard-band condition is indicated. It can be seen that in order to meet bus access time requirements or guard-band requirements, wait states must be inserted occasionally in the machine cycles of the processors. Thus, the arbiter circuit for this type of system must detect when a wait state is required in a processor machine cycle and put the processor in the wait state at the proper time.

We used simulation to investigate various 8085A multiprocessor systems in which the system bus is timeshared. Our model generates instructions for the processors and simulates the execution of those instructions by the processors. The model also simulates the bus arbitration circuits and their control of the processors and bus accesses. We compiled statistics for bus requests from, and bus grants to, the processors, and we recorded the simulation time. We used the IBM simulation language GPSS V.

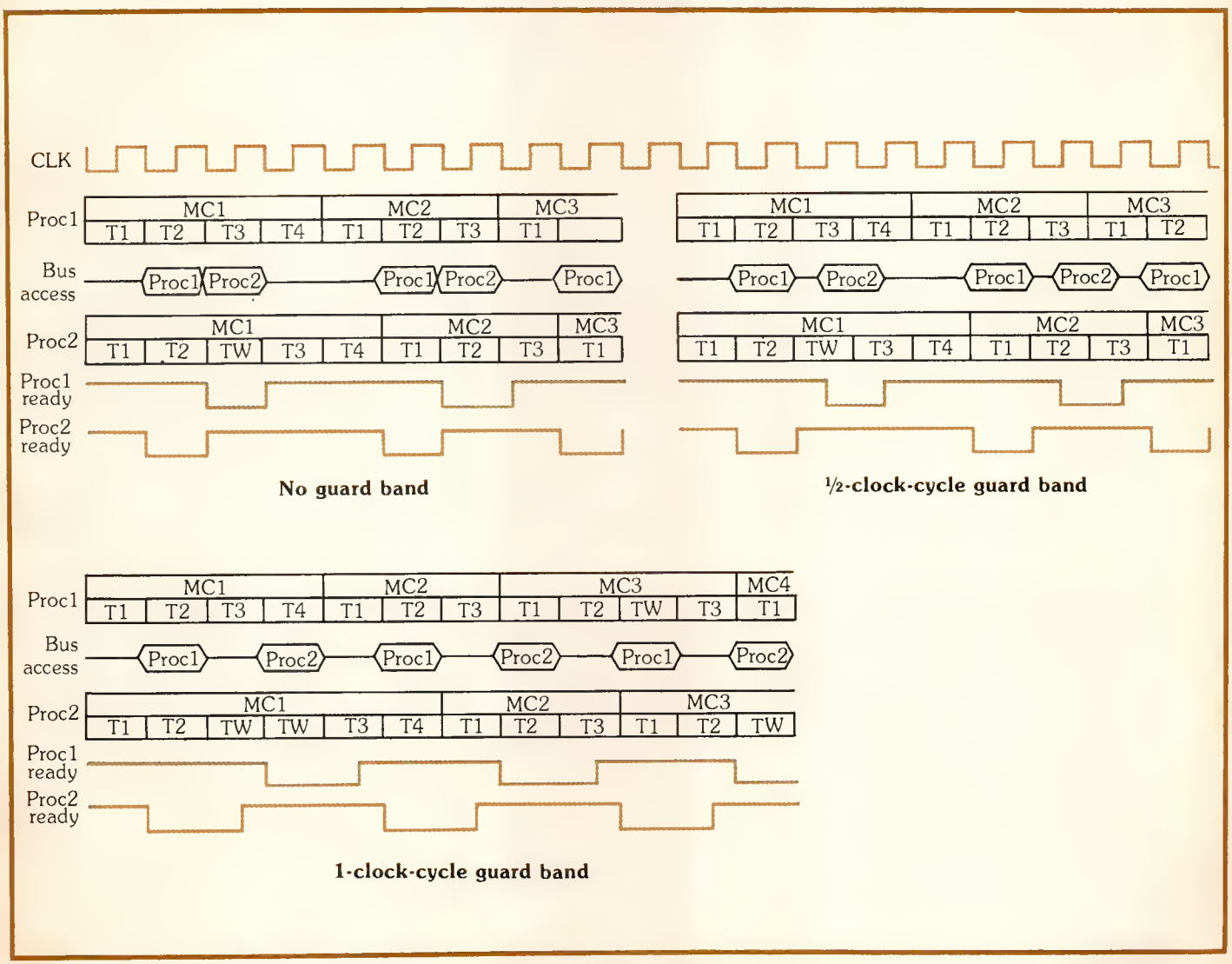


Figure 4. Timing diagrams for 1-clock-cycle bus access.

The GPSS model

The GPSS model (Figure 6) is divided into three segments corresponding to the three hardware sections shown in Figure 2. The first segment simulates the first processor, the second segment simulates the second processor, and the third segment simulates the bus arbitration circuits. The processor segments can be executed without the arbiter segment to

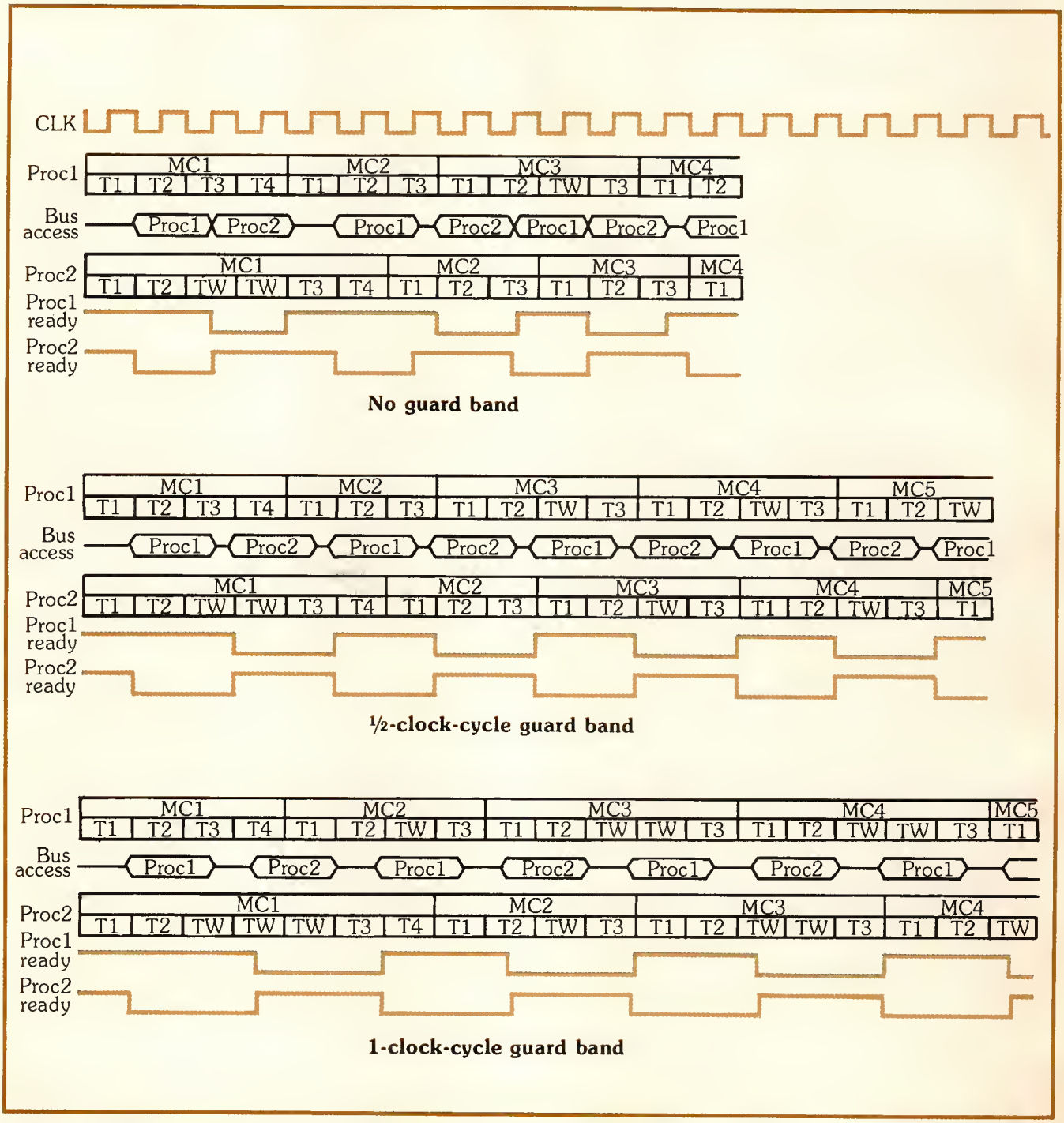


Figure 5. Timing diagrams for 1.5-clock-cycle bus access.

simulate independent operation of the processors without memory sharing. When the arbiter segment is executed with the processor segments, it controls them by causing one or both processors to enter a wait state for the required number of clock cycles.

GPSS allows the time period under study to be divided as finely as desired. The simulation then proceeds one time unit at a time, advancing each parallel process by the same increment. GPSS allows precise statistical control of events (transactions), using statistical data provided by the programmer. Table 1 shows the data that are used for generating the instructions to be executed by the simulated model. The instructions are grouped not by function but according to clock periods and machine cycles.

In our study, we were interested in the bus action during the operation of two processors sharing a common bus. Table 1 shows the instructions grouped according to bus action. To ensure that bus actions from each bus action type occurred, we based the statistics for the simulated instruction sequences on the percentage of the total instruction set having a particular bus action. These percentages are shown in Table 1. For instance, instruction type 4 is made up of instructions requiring nine clock periods and consisting of a six-clock-period instruction fetch and a memory read. In the 8085 instruction set, 6.8 percent of the instructions fall into this group. Therefore, in the simulation, 6.8 percent of the instructions are of this type.

Simulation results

In order to simulate both of the processors either executing the same instruction sequence or different instruction sequences, we developed two versions of the basic processor model. When the same instruction sequence is executed, the processor 1 model generates an instruction and sends an identical copy to the processor 2 model. The instructions are put into a queue for each processor. The list of instructions in each queue is identical and its elements are in the same order. The procesor models take instructions that are to be executed out of their assigned instruction queues. Because the contents of the queues are identical, the processors execute the same instruction sequence. A single random number generator is used to determine the instruction sequence. When different instruction sequences

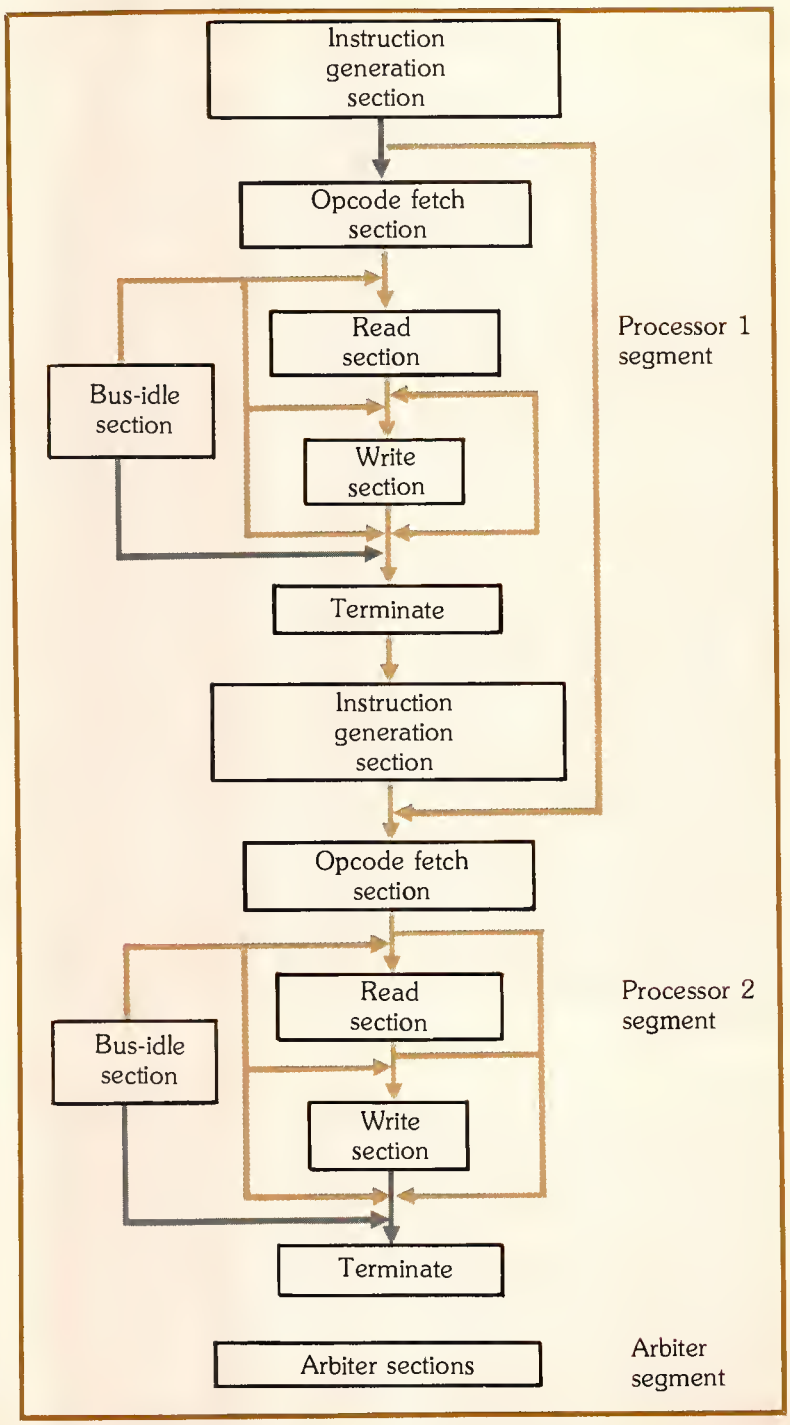


Figure 6. Flowchart for the GPSS model.

are executed, each processor model uses a separate random number generator to generate its instruction sequence.

For the simulations performed, certain assumptions were made. The simulation assumes that the processors are synchronous. In the multiprocessor system simulated, both processors run from the same clock and at the same frequency. Processor 1 is the master and has priority over processor 2. Upon power up, a wait state is automatically inserted in processor 2's first machine cycle to cause the required skew between the beginning of processor 1's machine cycle and processor 2's machine cycle. If both processors begin a machine cycle at the same time and request the system bus simultaneously, processor 1 is granted the bus first, and a wait state is inserted in processor 2's machine cycle. Processor 2 is granted the bus after processor 1 has completed its bus access.

Table 1.
Statistics used for assigning instruction types.

Instruction type	Clock periods	Clock cycle type, % occurrence	Machine cycle type		Machine cycle type, % occurrence
1	4	22.223	1	OP	22.2230
2	6	10.256	2	OP	10.2560
3	7	24.789	13	R	23.0765
			15	W	1.7095
4	9	6.838	23	R	6.8380
			177	I	.8548
5	10	15.385	135,136	R/W	3.4189
			133,134	R	11.1113
6	12	8.547	233	R	4.2735
			255	W	4.2735
7	13	1.709	1333	R	.8545
			1335	R/W	.8545
8	16	2.564	13333	R	.8547
			13355	R/W	1.7093
9	18	7.692	23355	R/W	7.6920

OP = Opcode fetch only
R = Read
W = Write
R/W = Read/write
I = Bus idle
1 = Four-clock-period opcode fetch
2 = Six-clock-period opcode fetch
3 = Memory read
4 = I/O read
5 = Memory write
6 = I/O write
7 = Bus idle

Table 2.
Simulation results for the single-processor system.

Instructions	BusA1	Avg. time/instr. (simulation units)
Sequence 1	39.2	1638
Sequence 2	39.4	1590
Average	39.3	1614

BusA1 = Percent bus was busy
Avg. time/instr. = Simulated time per instruction

Many statistics are computed by the GPSS program, such as the number of times the bus was requested and the number of wait cycles that were executed by each processor. The most important statistics obtained are the total simulated time and those associated with bus accesses by the processors. In our model, one clock cycle is equivalent to 200 units of simulated time. The simulation is set to run until a certain number of instructions have been executed. The amount of simulated time divided by the number of instructions executed gives the average amount of time required to execute each instruction. This average time per instruction is used to compare the efficiency of the multiprocessor system to that of a single-processor system.

Another important statistic obtained from the simulation is the percentage of total time that the system bus was used. Also provided are the percent of total time that the bus was granted to *each* processor and the percent of total time that the bus was granted to *either* processor. This last statistic is used to compare the bus-idle time of a multiprocessor system to that of a single-processor system.

The first simulations we ran were for a system with a single processor. We made a run for each of the two instruction sequences used in our model by setting each random number generator's seed number to the same value as that used in the multiprocessor simulation. We obtained statistics for the execution of 100 instructions, since only 100 instructions are executed by each processor in the multiprocessor model. The average time per instruction obtained with the two seed numbers was averaged and the result was used to evaluate the performance of the multiprocessor system

when its processors were executing different instruction sets. This established a baseline for use when the processors were executing different instruction sequences. Table 2 shows the results of the single-processor simulations.

The next set of simulations we ran was for the multiprocessor systems. As was stated earlier, the GPSS model is for a dual-processor system. We made twelve different runs for the execution of 200 instructions. The twelve runs represented every combination of instruction set, bus access length, and guard band length. The bus access length was either 1 clock cycle or 1.5 clock cycles. Table 3 shows the results of the multiprocessor simulations. The rows in each case are for guard bands of 0, 0.5, and 1 clock cycle.

For each case shown in Table 3, each processor executed 100 instructions out of the 200 simulated. A machine cycle of a particular processor was never extended by enough wait states to cause that processor to fall more than one instruction behind the other processor. The improvement over the single-processor system shown by the simulations ranged from a 36.7 percent to a 99.8 percent increase in instructions executed per simulated interval. The simulation showed that the improvement was greater when the two processors were executing the same instruction sequence. This gives an upper bound on improvement, since the processors are unlikely to execute the same instruction sequence except in special cases. More wait states must be inserted when the processors are not executing the same instruction sequence, since requests from the two processors occur at uneven intervals and sometimes at the same time. The improvement was greatest when there was no guard band and the bus access time was one clock cycle.

The simulation also showed that in all cases the utilization of the system bus was improved by the use of two processors. This improvement is a secondary goal of a multiprocessor system. The greatest utilization occurs when no guard bands are inserted and the bus access time is 1.5 clock cycles. Adding an interim register for each processor to hold the data from the end of T2 until those data are needed at the T3 midpoint makes bus utiliza-

Table 3.
Simulation results for the dual-processor system.

Case	BusA1	BusA2	Gra1	Gra2	GrBus	Avg. time/instr.	Speed up %
Same instructions	39.1	39.3	26.1	26.1	52.2	820	199.8
(1-clock-cycle bus access length)	39.1	39.3	26.1	26.1	52.2	820	199.8
	46.6	46.7	22.9	22.9	45.8	935	175.2
Different instructions	40.4	40.4	26.0	25.5	51.6	852	189.4
(1-clock-cycle bus access length)	40.4	40.4	26.0	25.5	51.6	852	189.4
	48.1	47.6	22.4	22.5	44.9	968	166.7
Same instructions	43.3	43.4	36.5	36.4	73.0	881	185.9
(1.5-clock-cycle bus access length)	46.6	46.7	34.3	34.3	68.7	935	175.2
	54.8	54.9	29.0	27.4	56.4	1106	148.1
Different instructions	43.9	46.0	36.5	34.6	71.1	923	174.9
(1.5-clock-cycle bus access length)	48.1	47.6	33.6	33.7	67.4	968	166.7
	57.0	56.0	28.1	28.1	56.3	1181	136.7

BusA*n* = The percentage of total time processor *n* requested the system bus.
Gra*n* = The percent of total time the system bus was granted to processor *n*.
GrBus = The percent of total time the system bus was granted to either processor (shows total use of system bus).
Avg. time/instr. = Average simulated time, in simulation units, required to execute each instruction.
Speed up = Baseline avg. time/instr. × 100 ÷ dual-processor avg. time/instr.

tion go down but the number of instructions executed per unit of time go up. Guard bands insert dead time on the system bus, and longer access times require use of the bus for longer periods of time. For practical realizations—in which a guard band is required—the bus utilization ranges from 44.9 percent to 68.7 percent. It appears that in our multiprocessor system, an additional processor could be added to use a portion of the time during which the system bus is not used.

We performed an additional set of simulations to determine the effects of nonsynchronous processor clocks on our dual-processor system. The system simulated is one which utilizes a bus access of one clock cycle and a guard band of one-half of a clock cycle to execute two different instruction sets. This system is a high-performance one which can be realized in hardware if sufficiently fast memory is available.

The simulation was performed with ¼-, ½-, and ¾-clock-cycle skew between processor 1 and processor 2. This was accomplished by delaying for a portion of a clock cycle the beginning of execution of the first instruction in processor 2.

The results for ¼-, ½-, and ¾-clock-cycle skew indicated that the average time per instruction remained aproximately the same for all three cases. In all three, one of the processors—processor 2—executed 101 of the 200 instructions. For ¼- and ½-clock-cycle skew, processor 1 was ahead of processor 2. For ¾-clock-cycle skew, processor 2 was ahead of processor 1. This was caused by additional wait states that had to be inserted in the machine cycles of the processors because of the clock skew. For ¼- and ½-clock cycle skew, additional wait states were inserted into processor 2's machine cycles. When the skew was increased to ¾ of a clock cycle, additional wait states were inserted into processor 1's machine cycles, which accounts for the fact that processor 2 executed more instructions.

Our simulated asynchronous multiprocessor system showed an improvement over the synchronous multiprocessor system of 79 to 88.7 percent more instructions executed per unit time. Though the improvements were less than those achieved when no clock skew was present, the degradation was less than one percent for ¼- and ½-clock-cycle skew, and 10 percent for ¾-clock-cycle skew. Additional data are required for a conclusive statement, but it appears that for asynchronous clocks the improvement lies in the bounds of the results obtained from the clock skew simulation results, and that a third processor can be effectively utilized.

We have shown how a GPSS model was used to simulate and evaluate dual-processor 8085A multiprocessor systems that employ the timesharing method of accessing the system bus and memory. The model was used to determine the effects of instruction execution, bus access length, guard band length, clock frequency, and clock synchronization on the operation of the multiprocessor systems.

For a multiprocessor system in which the processors ran from the same clock, timesharing always resulted in an improvement over a single-processor system as long as bus access length and guard band length were kept within reasonable limits. The number of instructions executed per unit time and the system bus utilization were increased. The greatest improvement occurred when the processors executed the same instruction set and the shortest possible guard band and bus access times were used in the system. The results indicated that a third processor could be added to the system and additional improvement attained.

For systems in which the processors do not run from synchronous clocks but do run at the same frequency, we performed a limited amount of simulation as a prelude to further study. The results indicated that even in a system in which the processors were asynchronous, timesharing of the bus resulted in an improvement. There was a degradation in performance compared to the synchronous system, but it was small. The results also indicated that when the clock skew was less than one-half of a clock cycle, the degradation was significantly less than when the clock skew was greater than one-half of a clock cycle. For the particular arbiter logic used, the amount of clock skew determined which of the two processors in the multiprocessor system executed

more instructions in a given amount of time. We expect that further simulation will verify if this is always true or if it is only a function of the instructions executed.

Our work suggests that

- further simulation and evaluation of non-synchronous multiprocessor systems should be carried out,
- multiprocessor systems in which the processors run at different speeds should be investigated,
- the simulation and evaluation of multiprocessor systems in which one processor is interrupt-driven and does not access the system bus continuously should be considered, and
- evaluation with actual programs should be performed.

References

1. S. K. Mehra, J. W. Wong, and J. C. Majithia, "A Comparative Study of Some Two-Processor Organizations," *IEEE Trans. Computers,* Vol. C-29, No. 1, Jan. 1980, pp. 44-49.
2. B. W. Arden and H. Lee, "A Regular Network for Multicomputer Systems," *IEEE Trans. Computers,* Vol. C-31, No. 1, Jan. 1982, pp. 60-69.
3. J. H. Patel, "Performance of Processor-Memory Interconnections for Multiprocessors," *IEEE Trans. Computers,* Vol. C-30, No. 10, Oct. 1981, pp. 771-780.
4. W. Y. Yue and R. P. Halverson, "Making the Most of Multiprocessing for Microcomputers," *Computer Design,* Feb. 1982, pp. 101-106.

Reader Interest Survey

Indicate your interest in this article by circling the appropriate number on the Reader Interest Card.

High 153 Medium 154 Low 155



Brian E. Corrigan III is a specialist engineer at the Boeing Military Airplane Company, Wichita, Kansas, where he has been involved in military avionics design and support equipment design since 1980. His interests include computer architecture, multiprocessor systems, microprocessor applications, MIL-STD-1553 hardware, and digital signal processing. A member of Tau Beta Pi and Eta Kappa Nu, he received the BS and MS degrees in electrical engineering from Wichita State University in 1980 and 1985, respectively.

Questions about this article can be directed to Johnson at the Dept. of Electrical Engineering, Wichita State University, Wichita, KS 67208.



Everett L. Johnson is an associate processor of electrical engineering at Wichita State University. Prior to joining the WSU faculty in 1971, he was employed by the Boeing Military Airplane Company in Wichita, Kansas, and by the Sandia Corporation in Albuquerque, New Mexico. His research interests include microprocessor applications, parallel processing, and the application of computers to aid the handicapped.

A senior member of the IEEE and a member of the IEEE Computer Society, Johnson has served as chairman of the IEEE Wichita Section and as chairman of the Wichita Sections of the IEEE Computer and Control Societies. He was cofounder of the IEEE Computer Society's Technical Committee on Computing and the Handicapped and served as its 1983-84 chirman. He has been a guest editor of *IEEE Micro.*

Johnson received the BSEE from the University of Kansas in 1962, the MSEE from the University of New Mexico in 1964, and the PhDEE from the University of Kansas in 1969.

A Microcomputer-based Control System for a Three-Joint Robot Arm

Eric H. Stelzer
International Business Machines

Randy H. Moss
University of Missouri—Rolla

This detailed investigation of the operating system and control program for a small model robot teaches lessons that the student can later apply in industry.

As industries become more and more automated in response to high labor costs and increased foreign competition, the demand for people trained in computer-aided design, computer-aided engineering, and computer-aided manufacturing intensifies. In many universities, the response to this demand is taking the form of more courses in CAE or CAD/CAM. This, in turn, creates a need for CAE/CAD/CAM laboratory equipment. Unfortunately, some pieces of this equipment can be quite expensive. One such piece is the industrial robot.

For research and training, a model robot system can take the place of an actual industrial robot, providing the same experience and capabilities. With such a model, a student can visualize abstract concepts such as motion and orientation in three dimensions, while a researcher can simulate industrial environments in which to demonstrate the results of his work.

A model robot system—or any robotic system, for that matter—can be considered to consist of three subsystems. Here, we will refer to them as the *physical system,* the *control system,* and the *operating system.*

The physical system comprises the frame, motors, joints, and whatever other electronic and mechanical devices are required to produce the desired function. The physical system can be thought of as a group of "links" (segments) connected by "joints" that are moved by "movers" (motors, for example). The inputs to the physical system are commands to the movers, such as ones to control motor speed. The outputs are the actual positions of the links.

The control system manipulates the physical system according to information passed to it from the operating system. The control system can be implemented entirely in hardware (e.g., an analog controller), but hardware costs can be reduced and more flexibility attained if a combination of hardware and software is used. The inputs to the control system are the desired characteristics of the physical system, such as the desired position of a joint and the desired velocity of a link, as well as the measured (actual) characteristics. Outputs are commands to the movers.

The operating system is the interface between the operator and the control and physical systems. It is almost entirely software. Inputs are commands from the operator. Outputs are the desired characteristics of the physical system.

Thus, the operating system and control system programs must work interactively, with the operating system communicating with the

0272-1732/85/0600-0022$01.00 © 1985 IEEE

operator and the control program with the robot. Though ultimately limited by the hardware, these programs defined the capabilities of the robot system.

Here, we present operating system and control system programs written for a three-joint robot arm controlled by an Apple II Plus microcomputer. We examine the types of commands used along with the calculations and bookkeeping required to support them, and we discuss how capabilities can be added to the system.

Hardware

The first step in defining the structure and capabilities of the operating system and control programs is to examine the hardware with which they must work. The programs can then be defined so as to take full advantage of the hardware's assets while coping with its limitations.

Our model robot is a three-joint, single-arm type (Figure 1). It was designed as part of a system simulating an industrial assembly line. A vision system recognizes small ferrous objects on a conveyor belt, and the robot, using an electromagnet on the end of its arm, sorts them. A block diagram of the hardware for the system is shown in Figure 2.

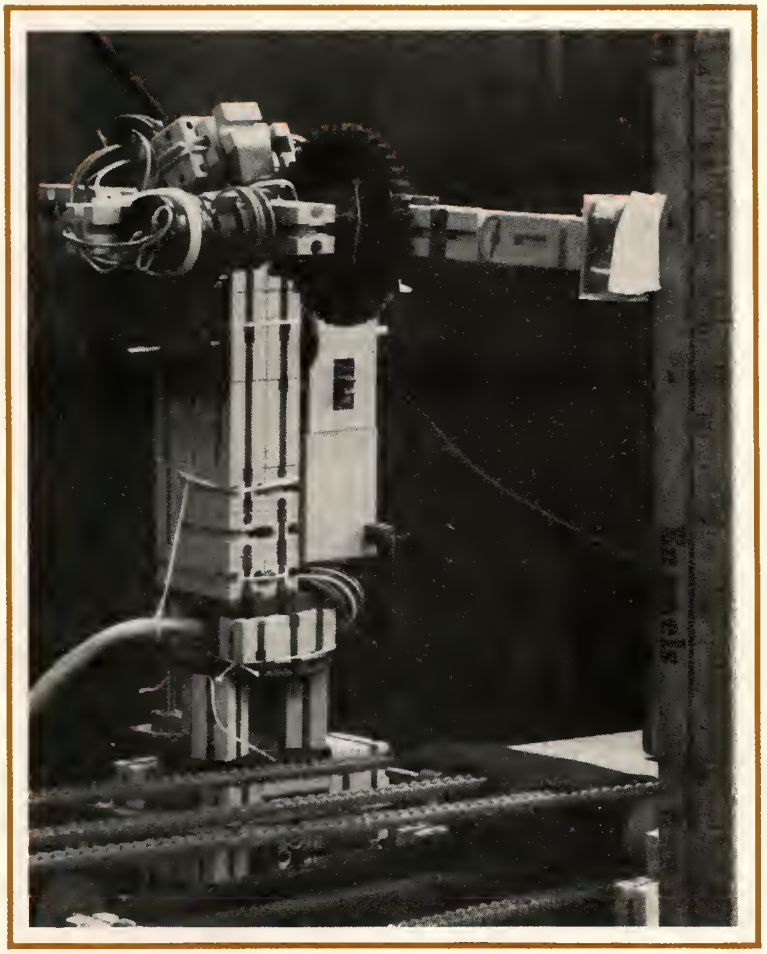


Figure 1. Three-joint robot arm in zero position.

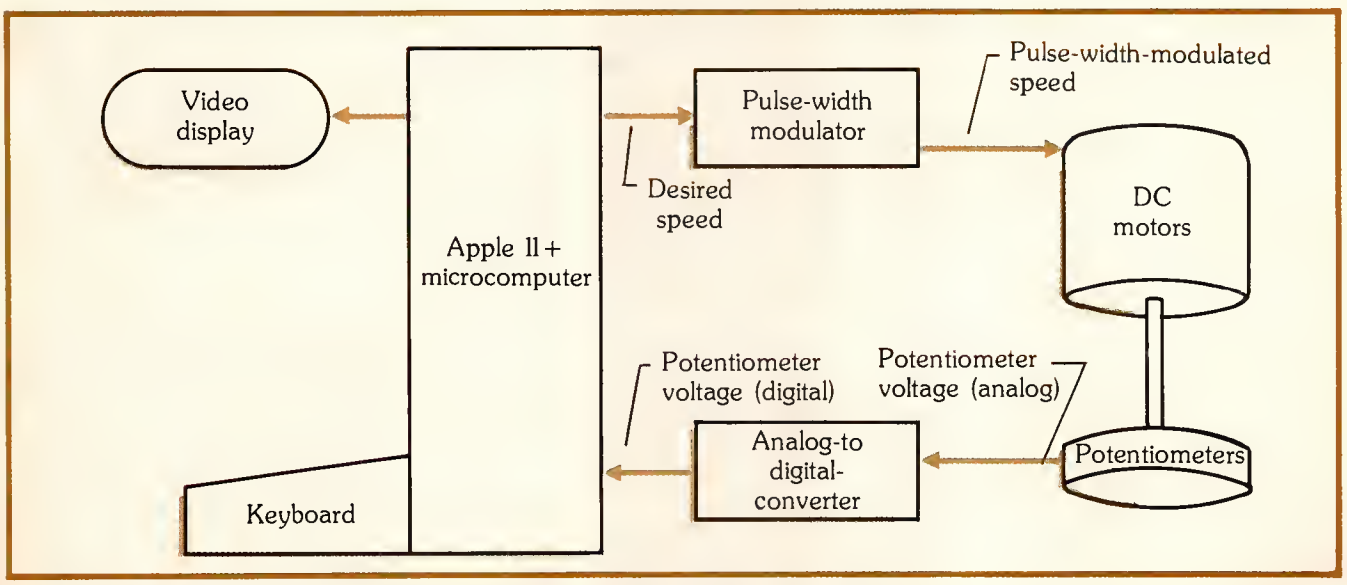


Figure 2. Hardware block diagram of the model robot system.

The DC motors that drive the joints of the robot are controlled by a pulse-width modulation system, which is in turn controlled by the Apple II Plus microcomputer. The model robot itself, the pulse-width modulation system, and the hardware interface to the Apple were designed and constructed by Richard Wainwright.[1,2] The interface to the Apple was designed so that each motor is configured as an output port. The desired speed for a motor is written to the output port and latched by the pulse-width modulator. The motor is then driven at that speed until another speed is written to the port. The speed input to the pulse-width modulator is a seven-bit binary number plus a forward/reverse bit—this scheme provides for a total of 128 speed settings in each direction. A speed of zero stops the motor.

The robot system commands can be considered an extension of Applesoft.

A switch on the pulse-width modulator system can disable the computer's control, allowing manual control of the robot's motors through switches and potentiometers mounted on the pulse-width modulator's front panel. An input port is provided in the Apple interface to allow the computer to read the status of this switch as one bit of the port. This port is referred to as the status port.

Single-turn potentiometers (pots) with no mechanical stops are mounted at each joint and read by the computer through an eight-bit analog-to-digital converter to provide position feedback. The voltage read on the pot is proportional to the rotational position of the pot, and thus to the position of the joint. An individual pot is read by writing to an output port assigned to that pot, which starts the conversion of its voltage into digital form. The output of the A/D converter is an eight-bit binary number (ranging from 0 to 255 in value) which can be read through an input port assigned to that task. Since the A/D converter takes a certain amount of time to perform the conversion, it uses an end-of-conversion (EOC) flag to tell the computer when it is finished. The computer reads the EOC flag as another bit of the status port.

If the pots on the joints were perfect, the voltage read from them would increase linearly from zero to the maximum and then immediately return to zero as the joint rotates. This is shown in Figure 3 (solid black line), where a reading of 128 (80 in hexadecimal notation) corresponds to a movement of 180 degrees from a reading of 0. Unfortunately, the pots are not perfect, and the actual readings (solid colored line) differ notably from the ideal. In this case, a reading of 128 does not correspond to a 180-degree rotation from a reading of 0. The software has to compensate for this.

The conveyor belt and the electromagnet can also be controlled by having the computer write the appropriate values to an output port assigned to each device. The conveyor belt's speed is pulse-width-modulated in the same way as that of the motors on the robot, and the electromagnet is turned on if the most significant bit of the number written to it is a one, and off if it is a zero.

Software

The software for controlling the model robot was written as two distinct programs: the robotics operating system (ROS) and the control system (CS). One of the objectives in writing these programs was to integrate their commands into the computer system so that the computer and the model robot could function as much as possible as a single system. Because of this integration, we will describe the Apple II Plus system along with the ROS's command parse routine.

The Apple system and the ROS command parser. The microcomputer system for which the ROS and CS were written is an Apple II Plus with a floppy disk drive and video monitor.[3,4] The Apple itself runs on a MOS Technology MCS6502 microprocessor chip.[5,6] To increase the speed of operation and provide access to as many special routines and locations in the Apple as possible, we wrote the ROS and CS in 6502 assembly language.[7]

Programming on the Apple is often done in a language called Applesoft Basic, which is a "standard" Basic with some commands for graphics and other special purposes added.[8,9] Most Applesoft commands can be executed in either of two modes:

- in *direct,* or *immediate,* mode, the command is entered through the keyboard of the Apple and executed as soon as a carriage return is typed;
- in *programmed,* or *deferred,* mode, a program composed of Applesoft commands is entered and then executed by the command RUN.

The disk drive is operated by a program supplied with the Apple II Plus called the Disk Operating System[4,10,11], which is loaded into memory from disk when the Apple is turned on. DOS commands operate in a fashion similar to Applesoft commands in direct mode, because when the DOS is loaded, the DOS command parse routine (the program which processes the input commands and either passes control to the proper subroutine or outputs syntax errors) is inserted before the Applesoft parse routine. However, in programmed mode this technique does not work, and the DOS commands must be embedded inside specially routed output statements.

The logical way to integrate the robot system's commands into the Apple system is to make them look and operate as much as possible like Applesoft commands, just as the DOS commands do in direct mode. In this way the robot system commands can be considered an extension of Applesoft. This objective is accomplished by inserting an instruction that jumps to the ROS command parse routine at the end of the DOS's command parse routine, and by including an instruction at the end of the ROS command parse routine to jump to the Applesoft command parse routine.

With this modification, a command typed on the Apple keyboard in direct mode enters the DOS parse routine. If not recognized as valid by the DOS, the command enters the ROS parse routine. If not recognized as valid by the ROS, the command is passed to Applesoft. And if Applesoft does not consider the command valid, it outputs a SYNTAX ERROR message.

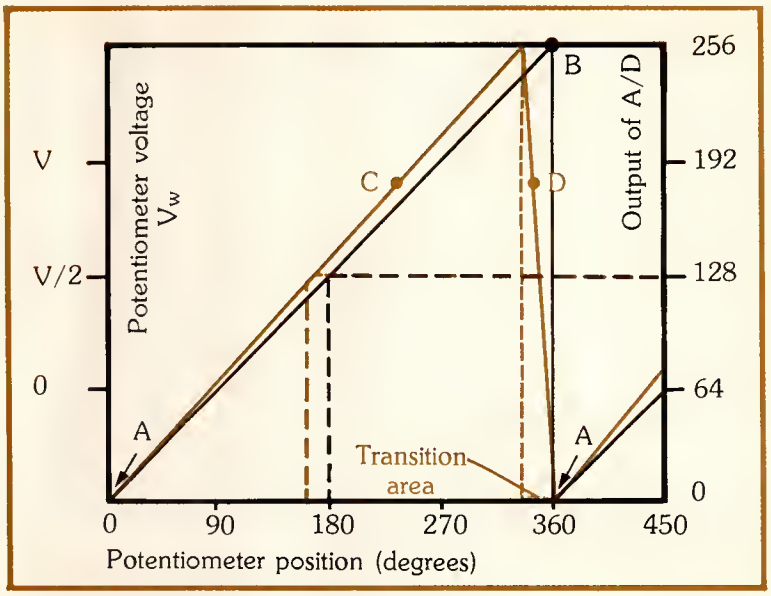


Figure 3. Ideal feedback potentiometer response (black) vs. actual feedback potentiometer response (color).

We could have designed the system so that in programmed mode the robot system commands would have been executed via the output statements used for DOS commands. However, a less awkward alternative was available. Applesoft has built into it an "ampersand trap" feature that branches to a special location in memory whenever the first character in a command is an ampersand. The location branched to is defined by a pointer in memory. Thus, setting this pointer to the ROS parser and prefacing the commands with ampersands enable robotic system commands to be used within an Applesoft program.

The final step in integrating the robotic system into the Apple is to make the loading of the ROS and CS programs, and the altering of the existing system, as transparent as possible to the user. When the Apple system is turned on with a disk in its disk drive, a program on that disk (usually called HELLO) is automatically run as soon as the DOS is loaded. This program is altered on the robotic system disks by the addition of commands to load the ROS and CS and make the necessary patches to the DOS command parse routine and the ampersand trap pointer. Then, merely turning on or reinitializing the Apple with this disk in the drive will cause the robotic system to be installed without the operator having to

type any special commands. Except for the memory space taken up by the ROS and CS and the robotics commands, the Apple system is left unchanged. A memory map is shown in Figure 4.

Reading the pots. The potentiometers on each joint of the arm are connected across a voltage source, and the wiper arm is fastened to the shaft of the joint. Thus, as the joint position changes, the voltage from the wiper arm to ground varies linearly from the voltage of the source to zero. The wiper has no mechanical stops and thus can rotate indefinitely in either direction. Ideally, this rotation results in a voltage V_w which changes with the position of the joint (black line in Figure 3).

This voltage is then input to an eight-bit analog-to-digital converter. The output of the A/D converter is thus a binary number between 0 and 255 that is proportional to the angular position of the joint. The position of the joint in degrees for this ideal pot can be determined by multiplying the output of the A/D converter by a scaling factor of 360/256.

The potentiometer may not necessarily be positioned so that a reading of 0 volts corresponds to the "zero" position of the joint (discussed later). In this case, a constant can be determined for the joint, and it can be added to the scaled output of the A/D converter. The modulo-360 remainder of the computed result yields the position of the joint in degrees.

Unfortunately, since the pots are not perfect, the transition of the wiper across the ends of the resistance element does not give the sharp break-off shown in black in Figure 3, but actually results in the voltage V_w shown in color in the figure. This causes problems if the joint is positioned in the transition area, since the A/D converter cannot distinguish between points such as C and D in the figure. Therefore this area of the pot must be avoided.

The robot has mechanical limits through which its "shoulder" and "elbow" joints cannot pass. By positioning the pots on the joints so that the transition zones of the pots are within these mechanical limits, we ensured that the transition zones would not be used. However, the "base" joint of the robot does not have any such mechanical limits. In this case, we positioned the transition zone of the pot in the least useful area of joint movement and defined limits for the joint in software so that this area could not be used.

At this point, however, the "good" zone of the pots still corresponds to a range of 0 to 255 on the A/D converter but only to one of 340 degrees on the joint. We compensated by changing the scaling factor from 360/256 to 340/256.

Path generation. When controlling the robot, the CS must determine where the robot is, and where it is supposed to be. When it has done this, the CS calculates the motor speeds required to move the robot into the desired position, and outputs this information to the pulse-width modulator. Finding out where the robot is presents no problem. The A/D converter can be read, and the reading can be processed as just described. Finding out where the robot is supposed to be depends on the path information passed from the ROS to the CS.

The format of this path information depends on how much time the CS can afford to take to process it. The simplest format to process is a list of points through which the robot is to pass—this list is evaluated at time intervals that correspond to the frequency with which the CS inputs information from the A/D converter. Thus, the CS is able to know which point in the list represents the current desired position. With this information, the CS can then determine the error in the joint positions. The main disadvantages of this format are that the list of points takes up quite a bit of memory space and that the list takes a lot of time to calculate.

The other alternative is to have the ROS generate equations for each joint that the CS can evaluate, knowing the current time, for each joint position. If the equation evaluation is implemented so as to reduce the computation time as much as possible, this alternative is still fast enough to allow the robot to be controlled in a satisfactory manner.

The equations that the ROS generates are in the form of fourth-order polynomials with constant coefficients:

$$p = c_4t^4 + c_3t^3 + c_2t^2 + c_1t + c_0.$$

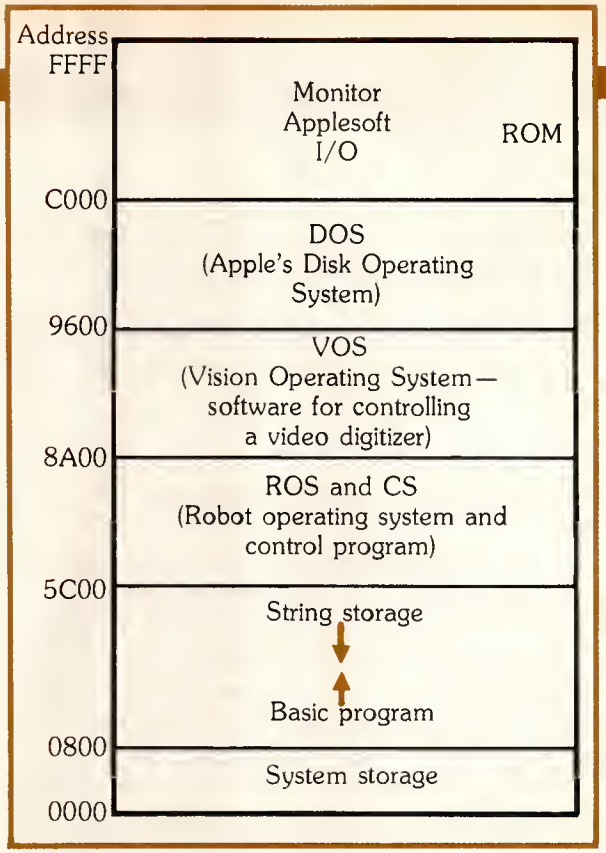


Figure 4. Apple system memory map.

Here, p is the position of the joint and t is time. The path information, then, consists of the coefficients of the polynomials for each joint. For complicated paths or paths which require moving through several points, several sets of equations are needed. Each set of coefficients making up one set of polynomials for each joint is called a *path segment*. The group of path segments that comprises a series of moves is called a *path set*.

There are several methods that can be used to move the end of a robot arm (called the end effector, manipulator, or tool-tip) from one point to another. In *direct motion,* each joint moves at full speed to its desired position independently of the other joints. In *joint-interpolated motion,* a proportional move is executed—the joint that will take the longest to get to its desired position is moved at full speed while the other joints are moved at speeds which will result in all joints reaching their desired positions at the same time. In *straight-line* motion, the joints are controlled so that the manipulator travels along a straight line at a constant velocity between the endpoints. Finally, in *continuous-path* motion, the manipulator is moved at a constant velocity through each point of a given sequence of points, with continuous acceleration of the joints.

The ROS supports joint-interpolated motion. The calculations for joint-interpolated motion can be made fast enough that no advantage is gained by implementing direct motion. The procedures needed to implement straight-line and continuous-path motion will be discussed later.

In the joint-interpolated method, the equations of motion for each joint are first-order polynomials of the form

$$p = vt + c,$$

where p is the position of the joint, v is the velocity of the joint, c is the starting point of the path, and t is time. The equations are calculated from the positions of the joints at the endpoints of each path segment.

The coefficients of these equations are stored by the ROS after calculation in an area of memory called a *path buffer*. There are five path buffers, each of which is capable of holding the coefficients of up to 45 fourth-order polynomials, or 15 path segments.

Path execution. As can be seen from the above, the CS, when it is enabled, looks into the appropriate path buffer for the coefficients of the polynomials that describe the joint motion desired. In order to evaluate these equations, though, the CS must have some way of keeping track of the time.

This is accomplished by providing a real-time clock which controls the frequency at which the CS does its calculations. The clock interrupts the computer at set intervals, and the CS is implemented as an interrupt service routine.

When an interrupt occurs, the CS first reads a location in memory called the *control byte*. If the least significant bit of the control byte is a one, the CS reads another memory location to determine which path buffer it is using, reads the coefficients and does the processing it needs to do to control the robot arm, and then returns control to the computer. If the least signficant bit of the control byte is a zero, the CS returns control to the computer without trying to control the robot arm. Thus, the ROS can turn the CS "on" or "off" by storing the appropriate quantity in the control byte.

With the CS implemented in this fashion, the computer can do its own processing, or commands can be entered and executed while the CS is moving the robot. The details of the method which the CS uses to control the robot will be discussed later.

Coordinate transformations

As discussed previously, the equations of a move segment are calculated from the positions of the joints at the endpoints. To allow the operator to enter an endpoint of a desired path as a set of cartesian coordinates, equations that can tranform these coordinates into joint positions must be provided. Since it would be convenient to also allow the operator to request the cartesian coordinates of the current position of the manipulator, equations that can perform the opposite transformation are needed.

The latter equations are called the *kinematic equations* of the robot model. We will discuss them first. The first-mentioned equations are the *inverse* or *solution of the kinematic equations;* we will discuss them after we deal with the kinematic equations.

The kinematic equations. As developed by Denavit and Hartenberg,[12] the relationship of a manipulator on one end of an arm to a coordinate frame at the base of the arm can be specified by a set of homogeneous transformations called A matrices. In this method, a 4×4 matrix is specified for each link of the arm; each matrix relates the position and orientation of the coordinate frame at the end of the link to the position and orientation of the coordinate frame at the base of the link. Thus, the transformation which relates the manipulator to the base of the arm can be obtained as a combination of A matrices.

Specifically, the transformation T_k, where k is the number of links in the arm, is given by the matrix product

$$T_k = A_1 A_2 A_3 \ldots A_k,$$

where A_k is the A matrix describing the kth link transformation. T_k defines the position and orientation of the manipulator, where

$$T_k = \begin{bmatrix} n_x & o_x & a_x & x \\ n_y & o_y & a_y & y \\ n_z & o_z & a_z & z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The kinematic equations

$$x = C_1(12C_{23} + 9.5C_2 + 1.5) + 3S_1 \quad \text{(Equation 1)}$$

$$y = S_1(12C_{23} + 9.5C_2 + 1.5) - 3C_1 \quad \text{(Equation 2)}$$

$$z = -(12S_{23} + 9.5S_2) \quad \text{(Equation 3)}$$

Solution of the kinematic equations

$$\theta_1 = 2\tan^{-1}\left(\frac{-x \pm \sqrt{x^2 - (y + 3)(y - 3)}}{y - 3}\right) \quad \text{(Equation 4)}$$

$$\theta_2 = \tan^{-1}\left(\frac{-12[C_3z + S_3(C_1x + S_1y - 1.5)] - 9.5z}{12[C_3(C_1x + S_1y - 1.5) - S_3z] + 9.5(C_1x + S_1y - 1.5)}\right) \quad \text{(Equation 5)}$$

$$\theta_3 = \tan^{-1}\left(\frac{\pm \sqrt{51984 - [(C_1x + S_1y - 1.5)^2 + z^2 - 234.25]^2}}{(C_1x + S_1y - 1.5)^2 + z^2 - 234.25}\right) \quad \text{(Equation 6)}$$

The vectors n, o, and a are mutually perpendicular unit vectors that describe the orientation of the manipulator. The coordinates x, y, and z specify the position of the manipulator in the coordinates of the base frame. By deriving the A matrices for each link and leaving a variable in the matrix to represent the link variable (the parameter that changes as the link is moved), one can obtain the equations relating the position of the manipulator to the base of the arm as the relations for x, y, and z resulting after multiplying the A matrices together.

Stelzer performs this calculation for the model robot in his thesis,[7] using the techniques developed by Paul.[13] The resulting equations for the model robot are shown at left. Here x, y, and z represent the position of the manipulator in centimeters, $C_k = \cos(\theta_k)$, $S_k = \sin(\theta_k)$, $C_{jk} = \cos(\theta_j + \theta_k)$, $S_{jk} = \sin(\theta_j + \theta_k)$, and θ_1, θ_2, and θ_3 are the positions of the joints of the model in degrees or radians, referenced to the zero positions.

Solution of the kinematic equations. In order to be able to find the joint positions required to place the manipulator at a certain cartesian location, we need to obtain the inverse of the equations derived above. Again, Stelzer performs this calculation in his thesis,[7] using Paul's techniques.[13] The resulting equations are shown at left.

The robotics operating system

The ROS-command-parsing method described earlier allows the operator to use the ROS commands as if they were actual Applesoft commands.

The commands used by the ROS must possess the following four characteristics:

(1) they must be easy to learn and use;
(2) they must be error-tolerant;
(3) they must protect the robot model from colliding with itself as much as possible; and,
(4) except as provided for in item 3, they must limit the capabilities of the hardware as little as possible.

The third characteristic is provided by the control program and will be discussed later. The others are provided through the ROS commands defined here.

The ROS commands can be divided into three groups. The first group consists of the various forms of the MOVE command, which generates and executes the paths along which the robot is moved. The second group of commands aids the MOVE command in the generation and execution of these paths. The commands in this group will be referred to as the move-related commands. The third group of commands does not deal directly with the robot, but takes care of other aspects of the system. The commands in this group will be discussed under the heading of miscellaneous commands.

The syntax and use of the ROS commands are covered in detail by Stelzer.[7]

The MOVE command. To move the robot, a path set must be obtained and placed in one of the five path buffers. Then the control program (CS) must be informed that it should execute that path set. The MOVE command accomplishes this objective in one of several ways. In its various forms, MOVE may be used to

- calculate a move segment or group of move segments, store them in the path buffer, and execute them;
- load previously calculated groups of move segments from the disk into the path buffer and execute them;
- combine calculated and loaded move segments into one path buffer and execute them;
- load the path buffer as above without executing it; or
- execute the move segments in a path buffer without loading the buffer.

As shown in the flowchart in Figure 5, the MOVE command's first task is to check if the CS is already executing a path set. If it is, this implies that the robot is in motion. In this case, the message

PREVIOUS MOVE STILL IN PROGRESS

is displayed and the new move is abandoned. This prevents a path that is currently being

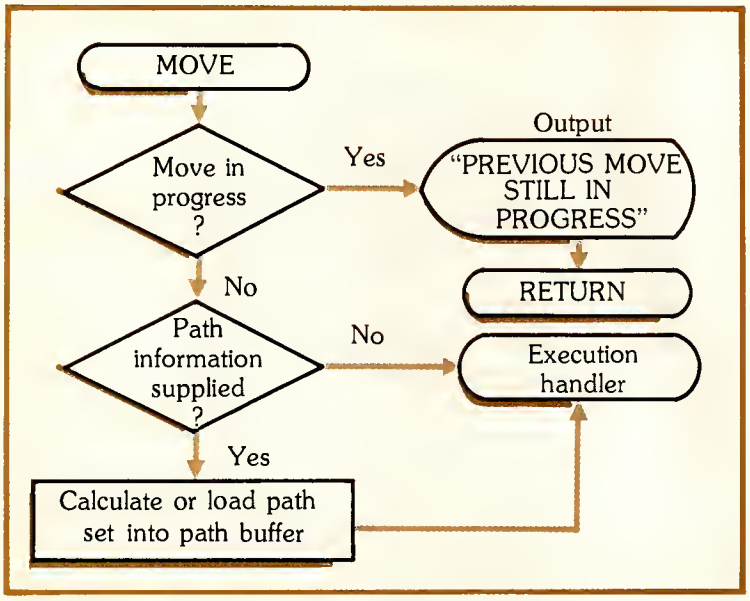


Figure 5. Flowchart of the MOVE routine.

used by the CS from being interrupted or changed before its normal termination.

The next step in MOVE command processing depends on whether or not information is supplied that allows the ROS to generate a path set to load into the path buffer before execution. If this information is not supplied, the ROS assumes that the path set is already in the path buffer, and it proceeds to transfer control to the execution handler.

As the flowchart in Figure 6 shows, the execution handler checks the status port to determine whether the pulse-width modulator is in manual mode or computer mode. If it is in manual mode, the CS will not be able to move the robot, and the message

ROBOT IS IN MANUAL MODE

is output and the move is abandoned.

If the pulse-width modulator is in computer mode, the MOVE command is checked to see if it specifies a path buffer. If it does not, the last path buffer that was specified for loading or execution is used. The path buffer is then checked to make sure that a valid path set is present. If one is not, the message

PATH EMPTY

is output and the move is abandoned.

If all is well so far, the current position of the robot is read, and a move at full speed from there to the first endpoint in the path buffer is calculated and inserted as the first path segment of the buffer. This step is needed because the robot may have been moved since the path buffer was loaded, in which case the starting point assumed by the buffer is not the actual position of the robot. Finally, the CS is told to execute the path buffer, and the ROS returns control to the Apple system.

If the information needed to generate a path set is furnished with the MOVE command, the ROS generates the path set and stores it in the path buffer specified at the end of the MOVE command. If no path buffer is specified, the path set is stored in path buffer 0.

The path generation information consists of data needed to calculate path segments, or of the names of disk files containing path segments previously calculated (with the LEARN command), or of both. If a path name is specified, the ROS looks for a disk file having that name and the extension .MVE. If it finds it, it loads it into the path buffer, leaving enough space in front of the loaded file to insert one path segment. The ROS then calculates a move at full speed from the last endpoint specified (or from the present position of the robot, if this is the first set of paths for the path buffer) to the starting point of the file.

For instance, the command

MOVE ("PICK"),("PLACE"),P2

causes the ROS to

(1) find the file PICK.MVE on the disk and load it into path buffer 2, leaving a space at the beginning of the buffer,
(2) calculate a move at full speed from the present position of the robot to the starting point of PICK and store the move in the spot left vacant in path buffer 2,
(3) find the file PLACE.MVE on the disk and load it into the buffer after PICK, again leaving a space,
(4) calculate a move at full speed from the endpoint of PICK to the starting point

of PLACE and store the move in the space left in the buffer for it, and
(5) transfer control to the execution handler.

Note that the execution handler recalculates the first path (repeating step 2) before turning the path set over to the CS for execution.

To calculate path segments, the coordinates of the endpoints of the segments can be entered either as cartesian coordinates or as joint angles. Cartesian coordinates are entered as three numbers, separated by commas, representing the x, y, and z coordinates of the endpoint in centimeters. The equations developed previously are then used to translate them into joint positions. The coordinates are checked to make sure they define a point within the robot's operating range. If they do not, the message

OUT OF RANGE

is output and the move is abandoned.

Joint positions are entered as three numbers, separated by commas, representing the position of the first (base) joint, of the second (shoulder) joint, and of the third (elbow) joint in degrees. This last number is followed by a comma and the letter J.

The path speed, which is the percentage of the maximum speed at which the path segment will be executed, is entered after the coordinates of the endpoint of the path segment that it affects. The path speed is entered as the letter S followed by an integer representing the percentage of full speed desired. If no speed is entered, full speed is assumed.

Using this information, the ROS calculates a path segment in the manner described earlier. Once calculated, the segment is loaded into the indicated path buffer, and the next path segment or disk file is processed and loaded. This continues until the path set is complete, at which point the path is executed.

A variation of the coordinate specification process allows coordinates to be taken from one of five buffers in memory instead of being explicitly specified in the MOVE command itself. These buffers are called coordinate buffers and are labeled V0 through V5. This variation is designed to let a program calculate the coordinates of the MOVE instead of having to choose between MOVEs with fixed coordinates.

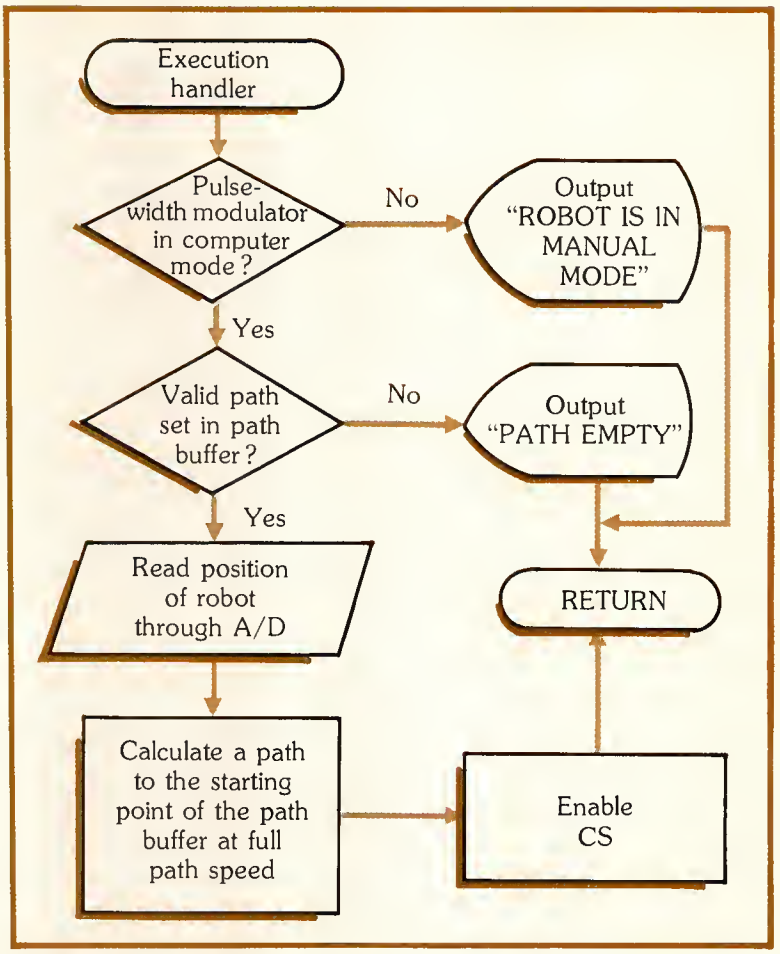


Figure 6. Flowchart of the execution handler.

The final type of path information that can be supplied with a MOVE command is the hold path. The hold path is a path which creates a delay at the present position. A hold is specified by H# in the MOVE command, where # is the hold time in tenths of seconds at a 100-percent execution speed (inversely proportional to the execution speed otherwise). The maximum value of the hold time is 600.

An example of these last two forms of the MOVE is the command

MOVE(V3,J),(H200),(V0,S30),

which

- moves to the joint positions specified in coordinate buffer 3,

- holds there for 20 seconds, and
- moves at 30-percent speed to the cartesian coordinates specified in coordinate buffer 0.

The final form of the MOVE command is the "deferred execution" form, specified by terminating the command with a W. This form of the MOVE operates exactly as the ordinary MOVE does, except that control is not passed to the execution handler after the path set is loaded into the path buffer. This type of command is used to load a path buffer for a future MOVE command.

The move-related commands. Although the MOVE command is very powerful by itself, it can be made even more effective and easier to use through the addition of the seven commands described below.

LEARN. A very popular method of programming industrial robots is to "teach" the robot by guiding the arm manually through the desired path, storing strategic points along the way. The path is recreated later by using these points as endpoints for path segments, much as the MOVE command does. The advantage of this method is that the operator can see the position of the robot without having to calculate the coordinates.

The LEARN command gives the operator the ability to do this with our model robot. The operator can also enter a set of coordinates or holds as he would in the MOVE command, if he desires. After the ROS learns a path set with this command, it stores it on disk and in path buffer 0. It can then execute the path set immediately by giving the command

MOVE,

or it can execute it later by using a MOVE command with the name of the file.

To use the LEARN command, the operator enters

LEARN "PATHNAME",

where PATHNAME.MVE is the name of the file in which the LEARN command stores the path set. The ROS then asks for the starting point of the path set by displaying the message

STARTING POINT?

If the pulse-width modulator is in manual mode when the RETURN key is pressed, the ROS assumes that the present position of the robot is to be used. If the pulse-width modulator is in computer mode, the ROS expects the coordinates of the point to have been entered from the keyboard. The format for the coordinates is the same as for the MOVE command.

If the operator wishes to enter the points by positioning the robot, he must switch the pulse-width modulator to manual mode. Then, by means of the switches on the pulse-width modulator, he positions the robot to the desired point and presses the RETURN key.

ROS then asks

NEXT POINT?

Again, the position of the switch upon RETURN is used to determine the source of the point data. If the path speed for the segment is to be other than 100 percent, it is entered also. Instead of an endpoint, a hold may be specified by entering H#, as in the MOVE command.

When RETURN is entered, the ROS inputs the data and again asks

NEXT POINT?

The process is then repeated for the next path segment.

When the path set is completed, the operator enters E after the ROS asks for the next input. The ROS then saves the file on disk and stores the path set in path buffer 0. To abort the path set without saving or storing it, the operator can enter X instead of E.

In computer mode, data for more than one path segment may be entered at once. In fact, if no manual data are needed, all of the information can be entered in one line, such as the command

LEARN "PICK",(0,0,0,J),
(15,5,−10,S75),(H400),(0,0,0,J,S90),E

S. The execution speed is the percentage of maximum speed at which the path set is

executed. The value of the execution speed is set at 50 percent upon initialization of the system, but it can be set to any integer value between 0 and 100 inclusive by the command S#, where # represents the integer value.

The actual speed of the MOVE will thus be the execution speed times the path speed, where both are percentages. Thus, in the command sequence

```
S75
MOVE(V0,S80)
```

the MOVE is executed at 60 percent of full speed.

W. In some situations, such as when MOVEs are executed from within a program, an indication is needed to let the operator (or the program, in this case) know that a MOVE is not currently active and that the next MOVE can be executed. The W instruction is a continuous check of the CS that does not return control to the operator until the CS is not executing a path. Thus, by executing a W before a MOVE command, the program avoids PREVIOUS MOVE STILL IN PROGRESS errors.

Arm configuration commands. When the manipulator position is specified in cartesian coordinates, there are usually four distinct sets of joint positions that will yield the same desired manipulator position. These four possibilities are shown for one point in Figures 7 through 10.

In Figures 7 and 8, the "shoulder" of the robot is on the right side of the base, and so this is called the "right-handed" configuration. In Figures 9 and 10, the robot is in the "left-handed" configuration.

In Figures 7 and 9 the "elbow" is positioned so that it points upward, and so this is referred to as the "elbow-up" configuration. Figures 8 and 10 depict the "elbow-down" configuration.

The different configurations are specified during the calculation of the joint positions from the cartesian positions by choosing either the plus or the minus in Equations 4 and 6. Choosing the plus in Equation 4 results in the right-handed configuration; choosing the

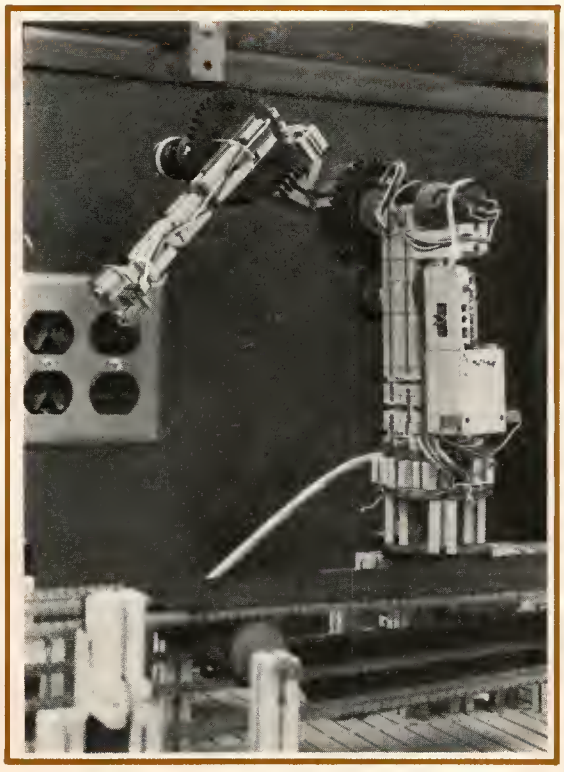




Figure 7. Right-handed, elbow-up configuration (left).

Figure 8. Right-handed, elbow-down configuration (right).

minus results in the left-handed one. Choosing the plus in Equation 6 results in elbow-up in the right-handed configuration and elbow-down in the left-handed one. Choosing the minus produces elbow-down in the right-handed configuration and elbow-up in the left-handed one.

After initialization, the system is set up to choose the right-handed, elbow-up configuration. This choice can be changed through the commands RIGHTY, LEFTY, ELBOW UP, and ELBOW DOWN.

Miscellaneous commands

CALIB. When a potentiometer is read to determine the position of a joint, a constant is added to "calibrate" the reading before the reading is converted into degrees. The CALIB command allows the operator to calibrate the robot by defining this constant for each joint.

The operator enters CALIB and the ROS then outputs the message

MOVE HAL INTO 'CALIBRATE' POSITION
ENTER 'RETURN' WHEN DONE

Using the controls on the pulse-width modulator in manual mode, the operator moves the robot into its "zero" position and then enters RETURN. The ROS reads the pots and sets the constants as the negatives of each pot reading.

CALIB can be inserted into the HELLO program on the robotics disks so that it is automatically executed (before any MOVE commands can be executed) after the robotics system is loaded into memory.

DISPLAY. It would help the operator if he could request information pertinent to the current state of the system. The DISPLAY command allows him to do so—it lists the following information on the Apple's monitor screen:

- the position of the robot in cartesian and joint coordinates,
- the execution speed,
- the last path buffer loaded or executed,
- the arm configuration (right-handed elbow-up, etc.), and
- the contents of each of the path buffers.

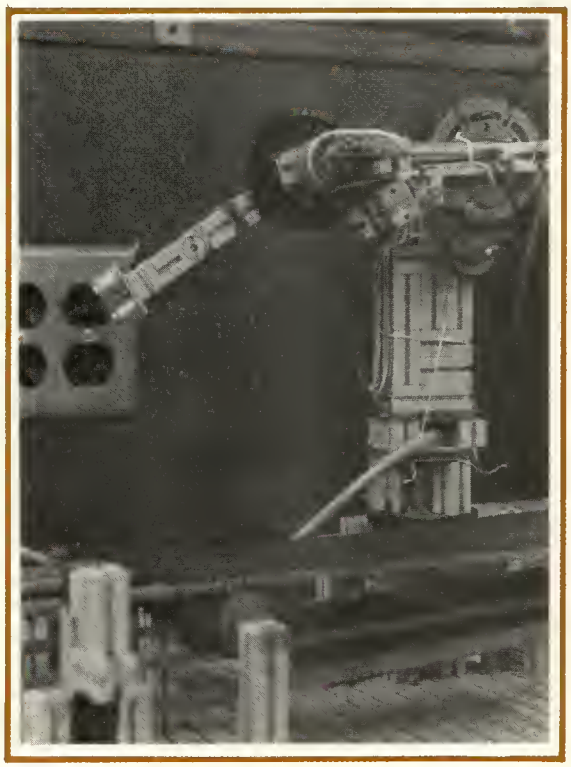


Figure 9. Left-handed, elbow-up configuration (left).

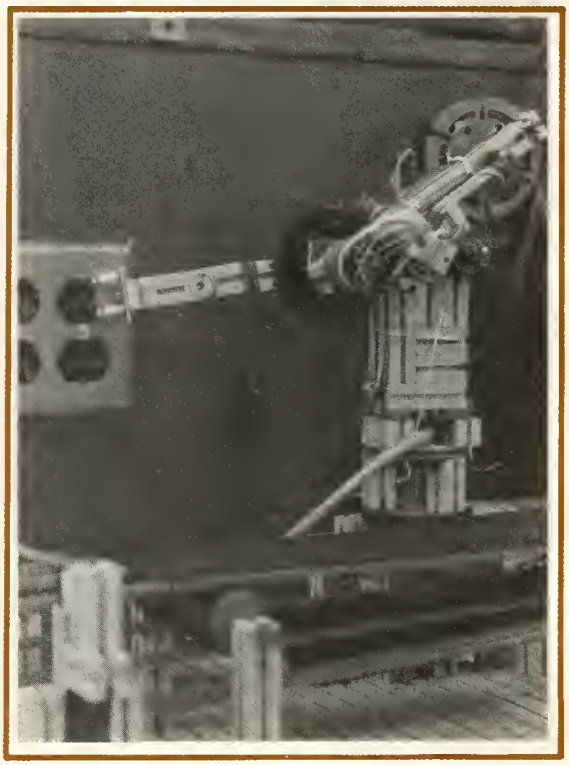


Figure 10. Left-handed, elbow-down configuration (right).

Electromagnet control commands. The electromagnet can be turned on and off by storing the appropriate number in an output port reserved for that purpose. This can be done with the Applesoft POKE command if the address of the port and the required value are known. The ROS commands PICK and DROP simplify this operation by storing the number in the port for the operator. PICK turns on the electromagnet. DROP turns it off.

BELT. As with the electromagnet, the speed and direction of the conveyor belt can also be controlled directly by the operator. However, this operation is simplified by the ROS command BELT(#), where # is an integer between −127 and 127 that represents the desired speed of the belt. Thus, the command BELT(127) turns the belt on at maximum speed in the forward direction (toward the robot). The command BELT(0) turns the belt off.

The control program

As explained earlier, the control program (CS) is implemented as an interrupt routine in order to control the robot efficiently in real time. Therefore, at presettable intervals the real-time clock requests an interrupt. The flowchart in Figure 11 shows the procedure followed by the CS to service this interrupt.

To avoid affecting the outcome of the interrupted processing, the CS—before it does anything else—must save all of the registers and memory locations that it shares with the normal processing routines so that it can restore them before returning control to the interrupted processing.

Both the ROS and CS have to be able to determine the present position of the robot—the ROS so that it can calculate paths and the CS so that it can control the joints during a move. However, since the ROS and CS read the A/D converter by first telling it which pot to read and then waiting for the conversion, they cannot both use the A/D. For instance, if the ROS tells the A/D converter to read a joint, and the CS interrupts and uses the A/D converter while ROS is waiting for the end of conversion, the result that the ROS finally reads from the A/D converter may not be the one it asked for. This dilemma is avoided by having the CS read the position of the robot every time an interrupt occurs and by storing the results in memory locations reserved for that purpose. The ROS can then access these memory locations when it needs to know the robot's position.

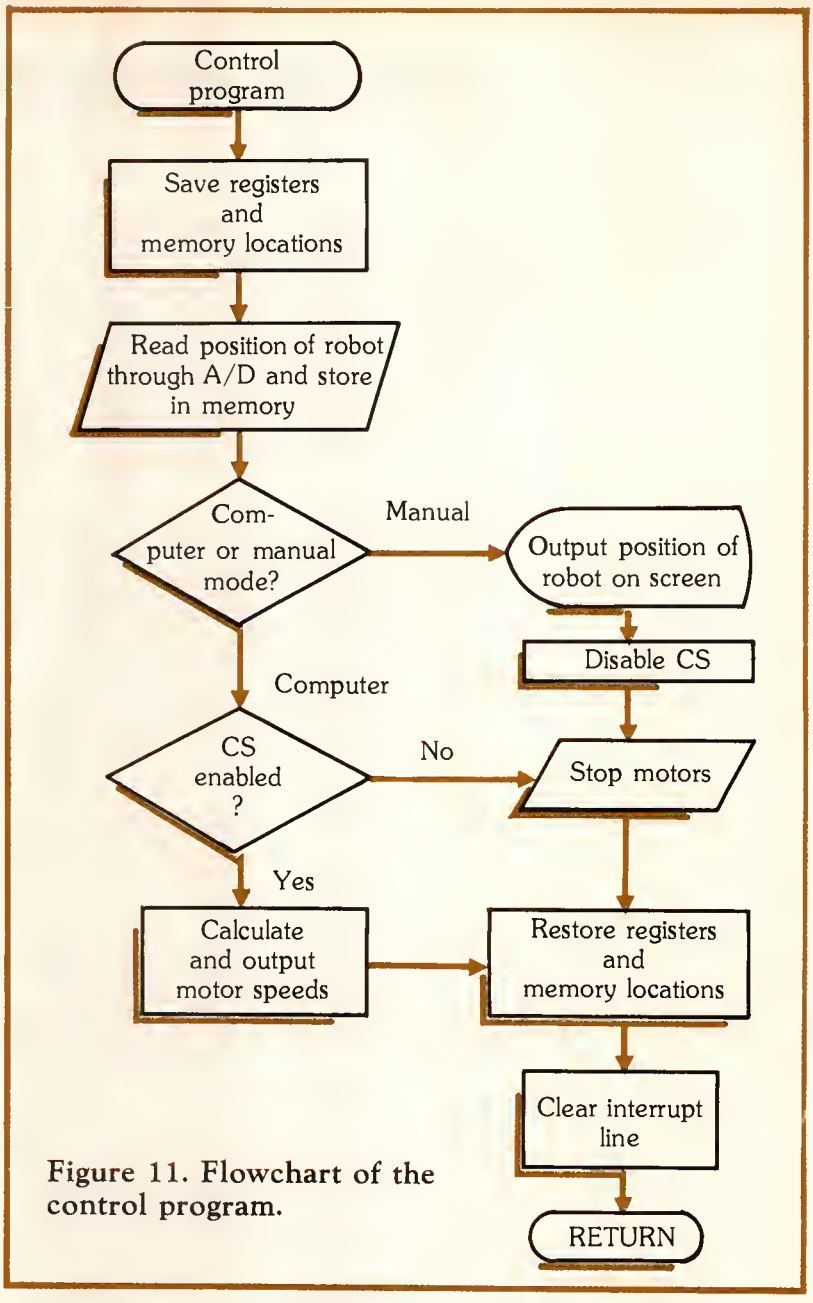


Figure 11. Flowchart of the control program.

After saving the registers and reading the position of the robot, the CS is ready for the next step, which depends on the current state of the system. By reading in the status port of the pulse-width modulator, the CS determines whether the pulse-width modulator is in computer or manual mode. If the pulse-width modulator is in computer mode, the CS checks the least significant bit of the control byte to

determine if the ROS has "enabled" it to execute a path. If not, the robot should not be moving, and the CS turns the motors off by writing a speed of zero to each motor port. Then the saved registers are restored, the interrupt line set by the clock is reset so that the clock can initiate an interrupt again later, and control is restored to the interrupted processing.

Controlling the joints. When the CS has been enabled to execute a path, the most significant bit of the control byte is checked to determine if the CS is executing another path. If this bit is a zero, there is no path being executed. In this case, the CS determines which path buffer is to be executed and sets its pointers to that buffer. Then the time counter, a two-byte memory location used to keep track of the real time, is set to zero and the starting points are calculated.

The value of the time counter should reflect the "real" time.

If the most significant bit of the control byte is one, the CS is executing another path. In this case, the time counter is compared to the maximum time count for that path segment. (A maximum time count is calculated for each path segment by the ROS and stored in the path buffer with the equation coefficients.) If they are equal, the end of the path segment has been reached. The pointers are then advanced to the next path segment of the path buffer, the time counter is set to zero, and the path equations are evaluated for the first point of the new path segment. If the time counter does not equal the maximum time count, a new value is calculated for the time counter before the path equations are evaluated.

Calculating the time. The value of the time counter is used in evaluating the equations for each path segment, and thus it should reflect the "real" time. Although this can be done by simply incrementing the value at every interrupt, another method is used so that the execution speed option discussed previously can be implemented. In this method, a constant equal to 2.55 times the execution speed percentage (specified using the S command) is added to an eight-bit fractional speed byte. The overflow from this modulo-256 addition is then added to the time counter.

Thus, if the execution speed is 100 percent, the time counter is incremented 255 times out of every 256 times that the addition is performed, or roughly every time. On the other hand, if the execution speed is 50 percent, the time counter is incremented 127 times out of 256, or roughly half of the times. The value of the time counter is therefore incremented in proportion to the execution speed, and thus the paths are executed at a speed proportional to the execution speed.

Evaluating the path equations. After determining the value of the time counter, the CS determines the desired positions for each of the three joints by reading the coefficients of the fourth-order polynomial from the path buffer and evaluating the equation

$$p = c_4t^4 + c_3t^3 + c_2t^2 + c_1t + c_0.$$

If this equation were evaluated as it stands, it would take ten multiplications and four additions. With Horner's scheme,[14] however, it can be rewritten as

$$p = \{[(c_4t + c_3)t + c_2]t + c_1\}t + c_0.$$

This equation can be implemented with four multiplications and four additions, and thus a substantial savings in execution time can be realized.

Calculation of motor speeds. Once the desired position of a joint has been determined, it can be compared to the actual position of the joint, and a motor speed can be calculated which will move the joint to the desired position. The simplest way to do this calculation is to subtract the actual position from the desired position, thus obtaining the position error, and then multiply by a gain constant.

This technique works if the error is large, but the gain constant must be very large for

the joint to respond to small errors. The reason for this is that due to the friction developed in the joints and gear trains, and due to gravitational effects in some positions, the joints will not move unless a certain pulse-width-modulator duty cycle is exceeded. We can compensate for this by adding a "base speed" that roughly corresponds to this starting pulse-width-modulator duty cycle to the result of the multiplication.

The response can be further enhanced by comparing the sign of the position error calculated for the joint during the previous interrupt to the sign of the present error. If they are the same, the CS assumes that the last motor speed output was not enough to compensate for the previous error, and it doubles the motor speed that was previously calculated.

This speed is then outputted to the motors through the pulse-width modulator. The saved registers are then restored, the interrupt line is cleared, and control is returned to the interrupted processing.

Although this control scheme is definitely not optimal, its calculation time is fast enough that the interrupt frequency can be set at 100 Hz. We decided not to attempt more sophisticated dynamic compensation because of the difficulty of achieving it in real time. By correcting itself 100 times a second, our control scheme works very well in actual operation. The motion observed at full speed appears quite smooth. At very slow speeds, the resolution of the eight-bit A/D converter appears as "clicks" as the robot moves from one setting to the next.

End of path control. Before resetting the pointers to the next path segment when the time counter reaches the maximum time count of the current path segment, the CS checks the maximum time count of the next path segment. If it is zero, the current path segment is the last segment of the path set. In this case, the pointers are not advanced and the time counter is not incremented. The desired positions of the joints (which are now the final positions of the path) and the motor speeds are calculated as before, but now the motor speeds that are output are monitored. When all of these speeds are zero for 10 interrupts in a row, the CS assumes that the robot has stopped at its final position and cancels the move.

Joint-error detection. In order to provide the robot with some degree of protection from self-collision, the CS compares the joint positions as it calculates them to a set of limits for each joint. When it detects a position that is out of the permissible range for a joint, it immediately stops the robot, cancels the move, and outputs the message

JOINT # *x* OUT OF RANGE,

where *x* is the number of the joint.

The limits of the "elbow" and "shoulder" joints are their physical limits. The limits of the "base" joint are determined by the placement of the transition zone of the potentiometer on that joint.

We were able to give our robot a degree of protection from self-collision.

Manual mode processing. When the operator is positioning the robot, he would probably find it advantageous to know the current position of the joints. Therefore, when in manual mode, the CS outputs on the top of the monitor screen the information

J1 = xxx J2 = xxx J3 = xxx,

where xxx is the position, in degrees, of each joint.

Since the robot cannot be controlled by the computer if the pulse-width modulator is in manual mode, the CS stores zeroes in the motor ports if the pulse-width modulator is in manual mode when an interrupt occurs. The CS also clears the control byte to cancel a move if one was in process. This allows the operator to use the computer/manual switch on the pulse-width modulator as a "panic button" to stop the robot, should trouble arise.

After clearing the control byte, the CS restores the saved registers, clears the interrupt line, and returns control to the interrupted processing.

Although the robotics operating and control system described here works well as it stands, improvements in both hardware and software can always be made. For instance, installing multiturn potentiometers and making the appropriate changes in the A/D converter would eliminate the need for limits on the "base" joint of the robot.

Improving the control system. As mentioned previously, the algorithm used to calculate the motor speeds in the control program is not optimal, although under most conditions it is satisfactory. However, there are situations in which the CS has difficulty controlling the arm in a smooth manner. While this problem could be caused by many factors, there are two that contribute significantly to it.

One is the fact that the gain that is used by the CS is constant for a given error, whereas the output to the pulse-width modulator that is required to correct the error depends on the current position of the robot. For example, the pulse-width-modulator duty cycle required to move the shoulder upward from the zero position (where it is pointing straight out from the base) is much greater than the duty cycle required to move it downward from the same position. Even worse, in this position these duty cycles change with the position of the elbow joint. Thus, to ensure that the robot will not "get stuck," the control algorithm must use a gain constant that results in a duty cycle large enough to move the robot in any direction from any position. This must be done at the expense of overcontrolling the joint when a position requiring a minimum duty cycle is encountered.

Compensating for this by making the gain a function of the positions of the joints is one alternative that could be used, although it would increase the CS's calculation time significantly, and require adjustment of the interrupt frequency. The speed of the Apple computer's microprocessor is a fundamental limitation here. Another alternative that would be less detrimental to calculation time would be to install strain gauges on each joint to provide torque feedback to the system. The gain could then be calculated as some simple function of the position error and of the torque on the joint. The torque feedback could also be used for such applications as collision detection and determination of the mass of a load.

The other problem with the present control system is the low resolution of the analog-to-digital converter. With an eight-bit output, it can represent up to 256 different positions for each joint. This corresponds to 1.4 degrees per "step," resulting in motion at slow speeds which seems rough. This could be corrected by installing a 12-bit A/D converter, which would provide a resolution of 0.088 degrees per step—more than enough for smooth motion.

The optimal solution to the problem of controlling the robot would be to dedicate a microprocessor to each joint; each microprocessor's only function would be to constantly monitor and correct the position of its joint. Path information could be transferred from the host computer (in this case, the Apple) to the microprocessors through shared memory space or through a FIFO buffer. This implementation would relieve the host computer of the interrupt routine, allowing it to process at full potential while the robot is running. This scheme could be taken one step further by having each microprocessor (or another microprocessor acting as an interpreter between the joint microprocessors and the Apple) calculate the paths, thus freeing the time and memory space used by the robotics system.

Alternate path motions. Continuous motion and straight-line motion would be useful, if not essential, to a robot designed to perform precision tasks such as spray painting and welding. We have made provisions to add these types of motion to our robotics system, but we have not yet implemented the algorithms required to do the calculations.

Continuous-path motion. A trajectory can be defined as a set of points through which the manipulator of the robot must pass, with predictable motion and no stopping between points. A smooth trajectory can be assured by specifying that position, velocity, and acceleration must all be continuous throughout the path. Continuous-path motion can be generated for a given set of points by express-

ing each path segment as a polynomial which satisfies these boundary conditions.

Ahlberg, Nilson, and Walsh[15] have shown that a curve-fitting method known as spline functions can be used to find the shortest path that satisfies the above conditions. Ho and Cook[16] have derived the equations needed to calculate the polynomials. Stelzer[7] has shown how the polynomial coefficients can be calculated for the model robot discussed here.

Straight-line motion. Straight-line manipulator motion could be accomplished by substituting linear equations using cartesian coordinates into Equations 4, 5, and 6. However, this quickly gets extremely messy. Moreover, the CS cannot handle equations in this form, so another method is needed.

As shown by Edwall,[17] straight-line motion may be approximated with the continuous-path scheme just discussed. The intended path is broken into smaller segments and a continuous path is calculated using these points. The more segments the path is broken into, the closer the result approximates a straight line.

Since continuous-path motion is already available, this implementation of straight-line motion is straightforward. However, care must be taken when using this scheme, since the path buffers will only hold enough information for 15 path segments.

Our model robot system is being used in conjunction with a vision system to identify objects on a conveyor belt, pick them up, and place them in designated bins. The robot system performs this complex task dependably. Therefore, our goal of implementing a robot system which can perform a specified function in real time has been attained.

Acknowledgments

The vision system used with our robot system consists of an Ikegami ITC-48 monochrome television camera, an Interactive Video Systems IVS-100 video processor, and software which runs on the Apple computer. The software for controlling the IVS-100 video processor was supplied by Interactive Video Systems. The pattern recognition software to perform the bin-sorting task was developed by Kun-Fuang Shen and is discussed in his thesis, "A Computer Vision System for Assembly Line Robots."[18]

When performing the bin-sorting function, the Apple operates at a point very close to its maximum capabilities in terms of both memory size and execution speed. In order to add more computational power to the system, Paul Braun designed and constructed an interface that allows an Intel single-board computer to be added to the system. More details about this interface can be found in his thesis, "Enhancement of a Robot Vision System with a Sixteen-Bit Microprocessor-Based Single-Board Computer."[19]

Reader Interest Survey

Indicate your interest in this article by circling the appropriate number on the Reader Interest Card.

High 156 Medium 157 Low 158

References

1. R. E. Wainwright and R. H. Moss, "A Microcomputer-based Model Robot System with Pulse-width Modulation Control," *IEEE Micro,* Vol. 5, No. 1, Feb. 1985, pp. 7-21.
2. R. E. Wainwright, "Microcomputer Control of a Model Robot System Using Programmable Pulse-Width Modulation," MS thesis, University of Missouri—Rolla, 1983.
3. *Apple II Reference Manual,* Apple Computer, Inc., Cupertino, CA, 1981.
4. W. F. Luebbert, *What's Where in the Apple—An Atlas to the Apple Computer,* Micro Ink, Inc., Chelmsford, MA, 1981.
5. *MCS6500 Microcomputer Family Hardware Manual,* MOS Technology, Inc., Norristown, PA, 1976.
6. L. A. Leventhal, *6502 Assembly Language Programming,* Osborne/McGraw-Hill, Berkeley, CA, 1979.

7. E. H. Stelzer, "A Control Program and Operating System for a Three-Joint Robot Arm on an Apple Microcomputer," MS thesis, University of Missouri—Rolla, 1982.
8. L. Poole, *Apple II User's Guide,* Osborne/McGraw-Hill, Berkeley, CA, 1981.
9. *Applesoft Basic Programming Reference Manual,* Apple Computer, Inc., Cupertino, CA, 1981.
10. *The DOS Manual—Apple II Disk Operating System,* Apple Computer, Inc., Cupertino, CA, 1981.
11. P. Lechner and D. Worth, *Beneath Apple DOS,* Quality Software, Chatsworth, CA, 1981.
12. J. Denavit and R. S. Hartenberg, "A Kinematic Notation for Lower Pair Mechanisms Based on Matrices," *J. Applied Mechanics,* Vol. 22, June 1955. (Also in *Trans. ASME,* Vol. 77, pp. 215-221.)
13. R. P. Paul, *Robot Manipulators: Mathematics, Programming, and Control,* MIT Press, Cambridge, MA, 1981, pp. 41-84.
14. J. F. Hart et al., *Computer Approximations,* John Wiley & Sons, New York, 1968, pp. 67-70.
15. J. H. Ahlberg, E. N. Nilson, and J. L. Walsh, *The Theory of Splines and Their Applications,* Academic Press, New York, 1967.
16. C. Y. Ho and C. C. Cook, "The Application of Spline Functions to Trajectory Generation for Computer-controlled Manipulators," *J. Digital Systems for Industrial Automation,* Vol. 1, No. 4, 1982, pp. 325-333.
17. C. W. Edwall, "A Trajectory Generation Method for a Computer-controlled Robot Using Spline Function Trajectories and the Implementation of a Control Program," MS thesis, University of Missouri—Rolla, 1982.
18. K. F. Shen, "A Computer Vision System for Assembly Line Robots," MS thesis, University of Missouri—Rolla, 1983.
19. P. A. Braun, "Enhancement of a Robot Vision System with a Sixteen-Bit Microprocessor-based Single-Board Computer," MS thesis, University of Missouri—Rolla, 1983.



Eric H. Stelzer is a senior associate engineer at the Industrial Systems Division of IBM in Boca Raton, Florida. At the time this article was written, he was a graduate research assistant at the University of Missouri—Rolla. Stelzer received the BS degree in electrical engineering, magna cum laude, in 1980, and the MS degree in electrical engineering in 1982, both from the University of Missouri—Rolla.



Randy H. Moss is an assistant professor of electrical engineering at the University of Missouri—Rolla. His research interests include pattern recognition, image processing, vision systems for industrial robots, digital systems design, and digital circuit design. He recently developed an image processing course and laboratory at the University of Missouri—Rolla under a grant from the General Electric Foundation.

Moss received the BS degree, with high honors, and the MS degree in electrical engineering from the University of Arkansas in 1975 and 1977, respectively. He was a National Merit Scholar and an NSF Graduate Fellow. He received the PhD degree in electrical engineering in 1981 from the University of Illinois. A member of Sigma Xi, Phi Kappa Phi, Eta Kappa Nu, and Tau Beta Pi, Moss served for two years as secretary-treasurer of the Rolla Subsection of the IEEE.

Questions about this article can be directed to Moss at the Department of Electrical Engineering, 116 Electrical Engineering Building, University of Missouri—Rolla, Rolla, MO 65401-0249.

Second-sourcing CPUs—Emulation, Ethics, and Electropolitics

Chuck Hastings

Competitive games aimed at "locking in" customers to particular hardware or software will ultimately fail. Serving customers, not manipulating them, yields market success.

In the traditional computer culture, CPU architectures and instruction sets have been regarded as property and CPU second-sourcing has been considered immoral, not to mention illegal and fattening. In the traditional semiconductor culture, on the other hand, nobody bought your fantastic new chip unless at least one of your hungry competitors second-sourced it, and so second-sourcing became a way of life.

A microprocessor is a CPU on a chip, and microprocessors and microcomputers have become big business. Hence, instant cultural collision! While semiconductor firms literally give their competitors masks and process data in order to persuade them to second-source their microprocessors, they get sued by minicomputer manufacturers for "pirating" and other lurid alleged crimes whenever they pick a minicomputer architecture, rather than a microprocessor architecture, to emulate. Meanwhile, on another front, there are at least a dozen firms around the world now second-sourcing the IBM 370 line of computers, at some level of "compatibility," and several others second-sourcing various members of either the DEC PDP-11 family or the Data General Nova family of minicomputers.

Is a manufacturer "stealing" if he emulates another manufacturer's CPU architecture rather than originating one through his own honest toil? Or, on the other hand, is he engaged in a devious attempt to "lock in" his customers if he invents yet another whole new instruction set or input/output interface incompatible with those of any of his competitors? Is a user "trespassing" if he runs software obtained from manufacturer A on a computer bought from manufacturer B who is second-sourcing A's machines without A's permission? Traditional legal and ethical theories haven't always shed much light on issues such as these.

If you don't first ask the right questions you normally don't get the right answers. This article attempts a fearless foray into some poorly mapped territory, asks some rather off-the-wall questions, and concludes with several sweeping opinions. If you think you may someday want to lay out some bread for a personal computer which emulates the instruction set of some existing minicomputer, you'll need to know this territory, so read on.

This article originally appeared in late 1978 in the *Conference Proceedings of the 3rd West Coast Computer Faire*. Paul Borrill, a past contributor to *IEEE Micro* who has been very active in IEEE standards activities, recommended that we reprint this essay. He felt, and the editors concurred, that Chuck Hastings' observations have remained timely and interesting. Chuck has added a postscript in which he comments on developments during the six years since his article first appeared.

We thank the Computer Faire and its founder, Jim C. Warren, Jr.

—*Ed.*

0272-1732/85/0600-0041$01.00 © 1985 IEEE

Property rights vs. civil rights

The idea for this paper burst upon me many years ago, when the regional sales manager for one of the major minicomputer manufacturers solemnly assured me—in answer to my routine question as to whether there was a second source for his company's one-board minicomputers—that second-sourcing another manufacturer's minicomputer was "immoral."



©1968, 1975 by Antonio Prohias and E. C. Publications, Inc.

"... as though we were from two totally different cultures ..."

He was quite startled when I let out a roar of laughter and uncontrollably broke up for several minutes. I was quite startled that my reaction should have surprised him. We then spent some time cautiously questioning each other, clarifying what we meant and each trying to understand where the other was coming from. It was as though we were from two totally different cultures suddenly thrown into collision—and perhaps we were!

This man thought of a CPU architecture as *property*. To duplicate this property—even in the broad sense of a reverse-engineered emulation—was, in his view, stealing. His company owned and controlled the property and was not about to permit such duplication. A cash customer who bought a minicomputer could enter upon the property—run programs—with his company's blessing. The idea that my employer really wanted to see some other company out there making almost-identical minicomputer boards, ones which would run the same programs as those which his company's boards ran, blew his mind—at least, at first.

I, on the other hand, was fresh from an attempt to evaluate about two dozen microprocessors, from as many companies. (This was about 1974, and many of those microprocessors never did finally make it out into the world as real products.) I was considering his company's one-board minicomputer as yet another alternative to committing my employer to going with some microprocessor which as yet existed only as an "objective specification" (i.e., a paper tiger) and developing a real-time microcomputer around that microprocessor, for a high-reliability application in providing local telephone service. My employer, like virtually all systems houses, had previously gotten burned by depending on sole sources for semiconductors and computer peripherals, and was twice shy about making future commitments to sole sources for anything. Moreover, it was particularly scary to commit to a microprocessor, an act which not only dictated a pinout and a part performance specification but also by implication rendered nontransportable a lot of expensive real-time software, when nobody was shipping parts yet and only one semiconductor house even had a full-scale development program underway for

a particular microprocessor. Thus, I took it as my employer's natural right to try to locate a second source for whatever microprocessor was selected and furthermore to run any and all software—whether we wrote it ourselves or acquired it by some other route—on the second-sourced CPU. In essence, I thought of running software on one's own CPU as a civil right akin to that of driving a rental car up one's own driveway, or having the "quiet enjoyment" of a leased residence.

In case you're wondering where this rambling anecdote may be leading, consider the current legal hassle between Data General and Fairchild Semiconductor. DG is probably the world's second-largest minicomputer manufacturer after DEC, and Fairchild is maybe the third or fourth largest semiconductor house. DG's low-end line is the famous Nova family, which the company has produced since it was founded. DG also has operated its own Silicon Valley semiconductor facility for several years, and has produced a considerable portion of what it has used, from TTL jellybeans to the one-chip NMOS MicroNova microprocessor around which it builds various microcomputer products. Fairchild chose the instruction set and architecture of one particular Nova model for its integrated-injection-logic, 16-bit, type 9440 "Flame" microprocessor, and somehow never got around to getting DG's permission. DG legally and loudly contested Fairchild's right to do so, and beyond that contested Fairchild's right to run any DG software on 9440-based microcomputers. Fairchild began advertising such microcomputers—as well as marketing the 9440—as semiconductor components for system builders, after some delay during which some Fairchild software got written just in case it turned out not to be cool to run DG software. Regardless of the final settlement of this case, it is best understood as a cultural collision, with each company genuinely convinced that right is on its side in accordance with the ethics of its culture.

Wry technical note: As a semiconductor component, the 9440 was not particularly compatible with the MicroNova. For one thing, it didn't need nearly as many different supply voltages. And as befitted a recently introduced bipolar part, it was apparently quite a bit faster.

DG's frame of mind, in deciding to go after Fairchild, may have had something to do with having sold a license several years before to another Bay-area company to use the Nova CPU architecture and existing software. The company was Rolm, now a major military computer and PBX manufacturer, and the pragmatic basis for the sale was that DG had no interest in selling militarized or ruggedized versions of the Nova and Rolm had no plans to sell any other kind. At the time that Rolm bought the license, it was a tiny start-up operation rather than a large and powerful company like Fairchild, and couldn't have afforded a strenuous legal hassle. However, after Rolm paid good money for a license to some things, it certainly strengthened the impression that these things were property, which is only a step away from the subsequent conviction that what Fairchild did was "stealing."

I took it as my employer's natural right to locate a second source for whatever microprocessor was selected and to run any software on it.

Embroiling of hapless third parties

What can happen to a user in a case like this one? On what may he count for reassurance that it won't happen?

If a user U buys a product from manufacturer B which appears to infringe a patent held by a competing manufacturer A, A can sometimes find legally tenable grounds for bringing suit against U as well as against B. This happened a couple years ago in one well-publicized lawsuit over modem patents. And it could even conceivably happen in a case like DG versus Fairchild—though the heart of the case may not concern patent rights, the minicomputer company may nonetheless attempt to go after third parties who use the semiconductor company's microprocessor in their products. Whether or not a legal action is well founded, it may serve well as a scare tactic to keep customers in the fold who might



Courtesy Winchester House

"... CPU architectures and instruction sets have been regarded as property ..."

otherwise switch to cheaper microcomputer boards which run the same instruction set as the minicomputer company's boards but are based on the microprocessor. It may even—who knows—scare off venture capitalists from investing in little companies which are trying to market such boards.

In another interesting situation several years back, effective reassurance was possible since the minicomputer company seemingly turned out not to "own the property" in question after all. By that time, various versions of the DEC PDP-8 minicomputer and its direct predecessors had sold in the tens of thousands of units. Intersil wanted to build a CMOS microprocessor and was looking for a CPU to emulate on silicon. They considered just two candidates—the PDP-8 and a popular 8-bit NMOS microprocessor. They chose the PDP-8 primarily because the resulting CMOS chip would actually be smaller and less complex than a CMOS chip derived from the other microprocessor, and because of the expected marketing advantages in having the 12-bit PDP-8 word length in a microprocessor. They apparently did not fully realize what a coup it would turn out to be, to be the first company to successfully emulate a popular minicomputer in a microprocessor. Thev did anticipate that DEC would not be altogether pleased with their sincere accolade to the popularity of the PDP-8, however, and accordingly did enough investigation to convince themselves that the PDP-8 architecture (as embodied in various predecessors) had essentially been developed at Lincoln Laboratories before DEC had even been founded, and therefore could be defended as something in the public domain. Even so, Intersil found it prudent to come to terms with DEC before purveying any DEC PDP-8 software to third parties who bought their IM6100 CMOS microprocessor. Having taken these steps, Intersil was in a position to convincingly reassure nervous customers who were apprehensive that they and Intersil would both get sued by DEC.

Total compatibility and the Fountain of Youth

It is often assumed that compatibility, like pregnancy, is a two-valued parameter—you is or you ain't. Wrongo. Compatibility has so many qualifiers, ins, outs, variations, and ramifications that it is most prudently regarded as a gray-scale or analog phenomenon. There really is no such thing as total compatibility—there is only compatibility for

some particular purpose such as running one particular piece of code or plugging in one particular peripheral device. "Total compatibility" is simply marketing hyperbole for compatibility for all practical purposes—all *practical* purposes—and can usually be proved to be embarrassingly nontotal by some bright rat fink with a pathologically clever application, which must then perforce be defined by the marketeer as impractical.

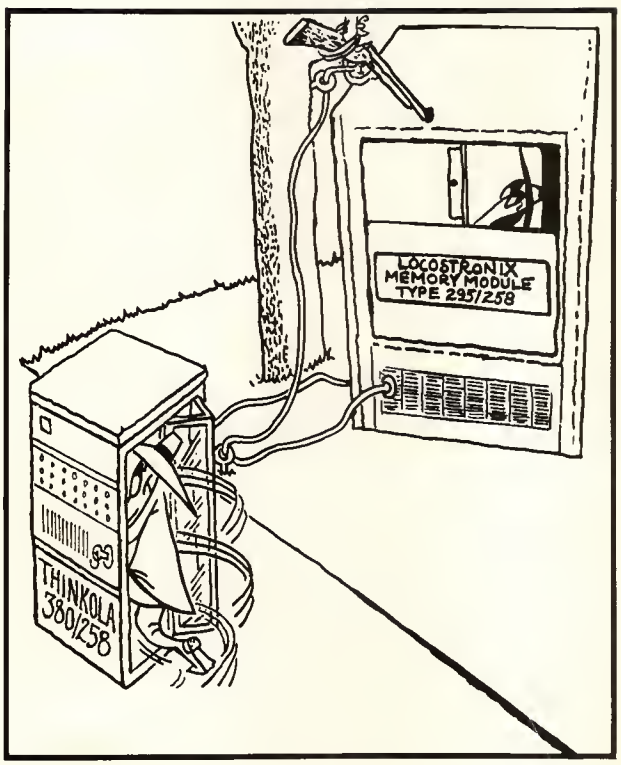
In case you think I am splitting hairs, consider the DEC PDP-11 family of computers. They all do considerably resemble one another with respect to such matters of practical interest as instruction sets and input/output interfaces. It is, however, relatively straightforward to code up a program using two-address instructions which refer to the same general register in both address-specifier fields, in one case using one of the auto-indexing addressing modes, and which will do a variety of different things on the various PDP-11 models, with departures particularly on the older models. To paraphrase George Orwell's famous line in *Animal Farm,* "All PDP-11s are compatible, but some are more compatible than others." If you want your code to be safely transportable from one PDP-11 model to another, study your DEC handbooks diligently to learn what the "impractical" cases are, and then avoid using them. There should always be some core set of features, which you can stick to, which operate with sufficiently good compatibility on all PDP-11 models in which you are interested.

Consider also the IBM 360 family of computers—a quite different case. Here, the manufacturer set up some tough company regulations to ensure both upwards and downwards compatibility (which I'll define in just a minute), since the different models were to be developed by administratively, and in some cases physically, separated groups of people. In essence, these various groups were each chartered to emulate the same target machine (apparently the middle-of-the-line 360/50), just as if they were all working for separate companies. An excellent degree of upwards and downwards compatibility was in fact achieved for the five models intended as mainstream—the 360/30, 360/40, 360/50, 360/65, and 360/75; some of the other models did have departures in various directions. However, even though the front door was locked, some incompatibility did creep in the back door in the form of optional features. On the 360/65 and larger machines, virtually all features with a serious impact on program transportability were standard, packaged-price items. On the smaller ones, however, one could pick and choose among various options in matters such as memory protection. In the 360/40 case in particular, there were eight or nine of these, with the result that a 360/40 could be ordered in about as many configurations as a hamburger from Wendy's, and code which accessed these optional features would not necessarily run on an economy-model, stripped 360/40 just because it ran fine on one loaded with extras.

Compatibility is best understood from a mathematical logic viewpoint. (PDP-11s, for instance, form a partially ordered set.) If a piece of code written for model 111 runs on model 222, we say that model 222 is *upwards-compatible* from model 111. Model 111 is then *downwards-compatible* from model 222 by an inverse definition, just as in formal mathematics subtraction is defined as that operation inverse to addition—strictly speaking, the statement A − B = C is first introduced as simply another way of stating that A = B + C. If, wonder of wonders, model 111 and model 222 are *both* upwards-compatible from each other, then they are *compatible*. However obvious these statements may seem here, what they imply—and, even more, what they fail to imply—is frequently ignored, with interesting and often expensive consequences.

"Total compatibility" is simply marketing hyperbole for compatibility for all practical *purposes.*

By the way, "compatibility" is probably the most often misspelled computer jargon word in the English language. One sees "compat*a*bility" three or four times as often as "compat*i*bility," even in print. Look it up in



©1968, 1975 by Antonio Prohias and E. C. Publications, Inc.

"Subtle changes in signal behavior at cable interfaces . . . however intended . . ."

the dictionary if you think that 40,000 computer gurus can't be wrong. The misspelling may possibly go back to an Ogden Nash one-liner of several decades ago about marital harmony, which goes something like "Incompatibility? No problem—as long as he has income and she is patable."

Add-on and add-in memories and CPUs

Many years ago, some bright entrepreneur noticed that most big mainframe CPUs were connected to their large, impressive memory modules through cables. Aha! If one could put a cheaper, denser, physically smaller, logically larger, more reliable "add-on" memory at the end of a cable having a similar connector, one could replace lots and lots of gigantic impressive memory modules made by the CPU manufacturer and make a lot of money. And this was done, most extensively with IBM 360 and 370 memories, but with others as well. In some cases, where the memory wasn't a separate cabinet but merely a module within the CPU cabinet, as in the General Electric 435, the second-source vendor, in this case Intel, provided an "add-in" rather than an "add-on" memory. Nowadays, an add-in memory may consist of but a single large circuit card, as does the Mostek add-in memory for the DG Eclipse CPU. Of course, the same business logic has been very well understood from the beginning on the personal computer scene.

It was also noticed long ago that large CPU peripheral devices, notably giant disk files, also connected to CPU input/output channels through cables with standard connectors. This provided clever entrepreneurs with yet another opportunity. When add-on peripherals became a lucrative business, the CPU manufacturers countered by such moves as pulling the disk controllers into the CPU cabinet so that the interlopers couldn't get to the cable interfaces anymore.

By then, however, some of the interlopers had themselves become large and powerful companies, and their success had inspired some well-financed new start-ups, and several of these companies attempted to carry the war back onto the enemy's territory by emulating the CPUs they were offering add-ons to. Although several of these attempts resulted in financial debacles which almost sank the interloper companies, the strategy ultimately prevailed, and today there are more than a dozen companies worldwide (including five communist-bloc state-owned companies) which essentially second-source several different IBM 370 models. Altogether these companies probably do several billion dollars worth of this business a year. A small fraction of IBM's annual sales, to be sure, but still a good piece of change.

White spy, black spy

As this second-sourcing business has grown large and successful, a fascinating set of electropolitical business strategies has developed along with it.

One obvious strategy is the pursuit of total compatibility. In one rather extreme case, a second-source house emulated an IBM CPU at the microcode level as well as at the instruction level! The rationale is that this capability will make it particularly easy to rapidly track changes made by IBM.

When a company whose CPUs and memory modules are being second-sourced desires to make engineering changes in their products, it may at times be very difficult to distinguish "our policy is one of continuous improvement" from a counterstrategy aimed at total compatibility. Subtle changes in signal behavior at cable interfaces and in operating-system behavior, however intended, can raise havoc in user installations having a mixture of first-source and second-source equipment, unless the second-source manufacturer has been able to anticipate and meet all of the changes. *Incompatibility thus becomes major competitive weapon*—fix things so that when your competitor hooks his box into your customer's system, the whole smash quits working and it's all your competitor's fault. The counter-counterstrategy to this was once described to me by the engineering vice-president of a major add-on memory firm as "You have to hang by a string inside an IBM CPU without disturbing anything."

Although the deliberate incompatibility strategy takes on some particularly interesting forms in the computing equipment business, it is not exactly unprecedented in human history. Take, for example, railroad track gauges. The standard American and British gauge is 4 feet, 8½ inches. This not-so-round number was derived in the surest possible way: A powerful major user (the Roman Empire) imposed it on other users (the Britons, whom they conquered a few years A.D.) by decree as a standard for *wagon* axle widths. The conforming standard wagons then wore standard ruts in British roads for many centuries, and users whose wagons had differing axle widths got a rough ride throughout those many centuries—or until they conformed. (Sound familiar?) Roughly two millennia later, the first British railway cars were fabricated simply as wagons with steel wheels, and so guess what the track gauge had to be? Elsewhere in Europe, however, many of the nonisland countries chose utterly unique gauges for their railroad rights-of-way, thus ensuring that any enemy troop trains which came charging across their borders would immediately jump the track (not to mention freight trains loaded with goods competing with those of the local merchants). Closer to home, when Cincinnati built its streetcar system around the turn of the century, a five-foot gauge was chosen so that the existing interurban railway company in the area could not run its cars over the streetcar right-of-way. The more things change, the more they stay the same.

Another intriguing old-time example having to do with the selective use of upwards compatibility concerns field artillery ammunition. Some years ago, the Russians chose to make theirs one millimeter larger in diameter than the existing western-world standard, and their gun barrels likewise. They were still able to shoot off captured enemy-manufactured shells with no real problems except perhaps for some presumed decrease in accuracy due to increased wobble of the shell in the gun barrel. On the other hand, enemy artillery units which attempted to make use of captured Russian shells were likely to find that these shells didn't quite make it all the way out of the gun barrel before coming to rest and blowing up.

In the present-day electronics world, similar games get played at the component level. For example, a large semiconductor manufacturer G may deliberately introduce a part similar to one from a smaller manufacturer H in an incompatible pinout, citing some rather unconvincing advantages, instead of simply second-sourcing it. The obvious intent here is to force systems houses using the part to lay out their boards to G's pinout, and thereby lock H's part out of all those sockets and H out of all those dollars. As often as not, however, this deliberate incompatibility game boomerangs—some other manufacturer second-sources the part in H's pinout, and G's part falls flat in the marketplace until G belatedly comes out with a part in H's pinout after all.

The *Mad* magazine Spy vs. Spy cartoon series by Antonio Prohias, excerpts from which appear in this text by permission (and with obvious modifications in one case), exuberantly

illustrates the wheels-within-wheels booby-trap philosophy which sometimes appears to prevail. However, in Spy vs. Spy, it is usually one of the protagonists who gets blown up. In the real-world second-sourced-CPU and add-on game, unfortunately, it isn't—rather, it is the customer who simply wants the most cost-effective computing equipment he can buy for his installation, from whatever vendor, and wants it all to play together reliably in place after he gets it.

Standards activities

Obviously, if instruction sets and interfaces were published in explicit, fully divulged form and not tinkered with after that, many of the devious games just alluded to would cease to be effective. Thus, users and second-source manufacturers share an increasingly intense interest in trying to standardize architectural attributes which affect them in some way.

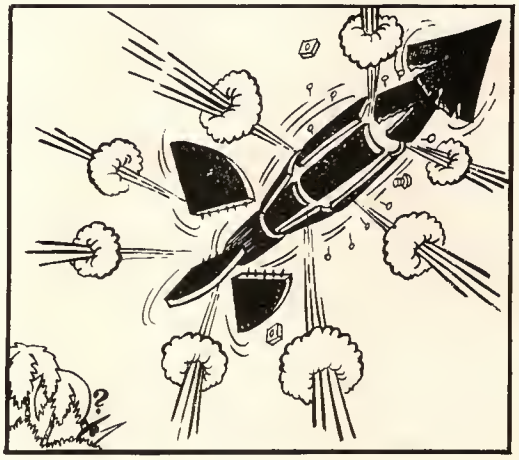
There are hazards to this approach also. The most obvious one is that premature imposition of standards in a technical area will freeze out worthwhile future improvements; there is a very legitimate aspect to "our policy is one of continuous improvement." But there is another one, probably more deadly in practice, and that is that standards inevitably operate to the benefit of some sets of economic interests and to the detriment of others. This eventuality is often foreseen, and the foreseers then dig in their heels and fight like hell to keep any standard from being adopted or to keep the one adopted toothless. But sometimes it is not altogether obvious who will be the beneficiaries and who will be the losers left out in the cold.

For instance, consider the move by various forces within the US government to require that all computer procurements over $400,000 in value make use of the IBM 370 input/output channel interface, supposedly to facilitate effective competition among add-on peripherals houses as well as between all of them and IBM. Honeywell loudly opposed this standard for obvious reasons, perceiving it as a Trojan horse rumbling through the city gates, and threatened to withdraw from competing for procurements subject to the standard rather than uproot its whole input/output architecture and replace it with the standard. So the standard would have represented a major cost for those manufacturers like Honeywell who were not compatible with it, unless they were to be locked out of, and their competitors locked into, sales of large government systems. But wait a minute—why was IBM also rather cool about the standard? Did they want to make some additional changes in the way their input/output channels behaved, which they wouldn't have been able to do and remain standard? If it wasn't IBM's Trojan horse, just whose was it? Possibly it was Amdahl's. Amdahl is the world's largest IBM 370 second-source manufacturer and vigorously supported the standard.

Thus it is difficult to avoid having *standards themselves become a competitive weapon,* like

"As often as not, however, this deliberate incompatibility game boomerangs . . ."



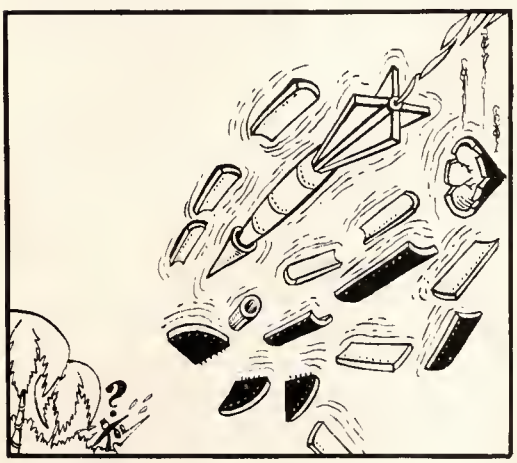


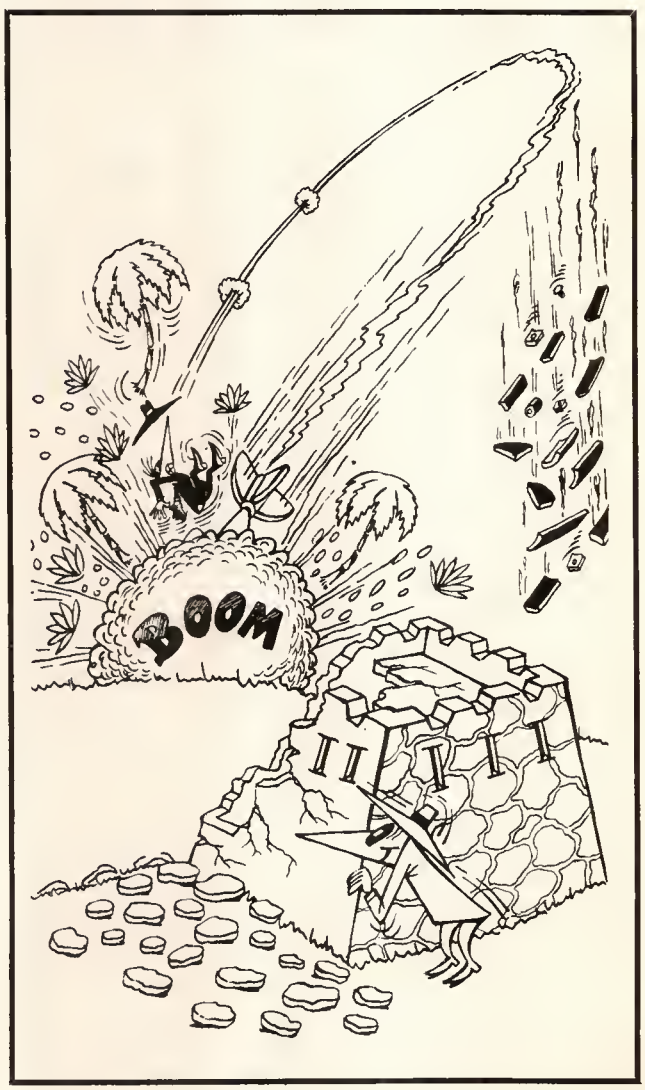
incompatibility. Get your format specified in, and your competitor's format specified out. When the protagonists are semiconductor firms, read "pinout" for "format."

Even so, the economic logic behind standardization is strong, and attempts to standardize go on. After all, once a standard has taken effect and has been in place for a while, the sharp and admittedly unfair immediate effects on some of the protagonists should die out and everyone should begin to benefit. Within the last year I have attended some meetings of two quite different standardization committees: one under the auspices of the US Army Electronics Command, which is attempting to arrive at a standard architecture for a military computer family, or MCF, and one under the auspices of the IEEE Computer Society, which is attempting to put forth a comprehensive standard for floating-point number formats and arithmetic in future microprocessors.

Both of these committees seem to have a substantial majority of people sincerely trying to do the right thing. Both of them, however, have tended to make haste very slowly. Any standard which either of them puts forth may tend to make life much easier for some of the agencies and economic interests represented and much more difficult for others, and large amounts of money and even the survival of companies—and thus the jobs of some committee members—may be at stake!

At one point, the MCF committee had managed to gather considerable momentum towards standardizing on the DEC PDP-11/70 instruction set and internal CPU architecture for future military minicomputers, along with cabinetry which bore at least some resemblance to that of the Control Data 480 military minicomputer. But there were loud screams from the military computer divisions of Litton and Westinghouse, who perceived themselves as potential recipients of a Trojan horse, and the almost-jelled consensus came unjelled. Whatever the recent trends may be, the final outcome is likely to be in doubt for several years. At best, the MCF architecture is likely to become yet one more flavor of military computer in addition to all the others in use, rather than a universal standard.





©1968, 1975 by Antonio Prohias and E. C. Publications, Inc.

Proposed standards can be beneficial; imposed standards will be disaster.

Compared to similar efforts that preceded it, the MCF activity for the most part has seemed to be sane and well-founded, with fairly modest goals and realistic notions as to what the world is actually like. The entire October 1977 issue of *Computer* (the general-interest magazine published by the IEEE Computer Society) was devoted to the MCF and is worth reading even if you have utterly no interest in khaki-colored computers. This is the *Computer* issue whose cover portrays heroic-looking eagles against a psychedelic color backdrop, and if you are at all familiar with some of the earlier military computer architectures which the MCF committee was quietly hoping to phase out, you may react as I did that a much better choice could have been made for the species of large native American bird pictured on the cover. Nevertheless, the articles in this issue range from good to excellent, and the first few have a direct bearing on the subject under discussion here.

The floating-point standards subcommittee is still hard at work, and the outcome of its efforts can't yet be predicted. The concepts being discussed at the time I write these words bear some resemblance both to those in DEC's VAX-11/780 supermini and to those in devices in development as part of Intel's 8086 program. However, unlike the MCF committee, this subcommittee is proposing to write a totally new standard which does not exactly correspond to any existing hardware, thus reducing the Trojan-horse aspects somewhat. By the time you read these words, some definitive action may have been taken.

The following six maxims, which are from an excellent article[1] which should be readily accessible to most of you, pertain to standards of almost any type as well as to the software standards about which the article was written:

- *Pro*posed standards can be very beneficial; *im*posed standards will be disaster.
- Standards are not only meaningless, they are actually *dangerous,* if their proper use cannot be understood by those affected by them.
- The most important part of a standard is its *applicability clause,* which exactly specifies when *not* to use it.
- Standard*ized* methods can become standard methods only when accepted in use.
- Standards exist only to *serve the needs* of a *non*standard world. Standards cannot create a standard world.
- *Compatibility* is more achievable than *conformability,* and usually is all that is needed.

Several sweeping opinions

It's always much easier to point to problems than to figure out what to do about them. Many of the problems I've described are inherent in any situation in which information-handling machinery is designed and marketed by human beings with something to gain from its widespread acceptance. They are not unique to one industry, one nation, or one economic system, and they will be with us for as long as we continue to use computers. However, my basic subject here is ethics, despite the unfamiliar context—what is it right that we *should* be trying to do? The particulars which I have been discussing seem to me to point to the following generalities:

- For manufacturers, whatever policies and decisions *serve* customers better are likely to work out best, both from an ethical viewpoint and for staying in business over the long haul. Customers are what make paydays and dividends possible. In a sufficiently long-term view (probably 3-5 years), there is virtually 100 percent correlation between sound marketing practices and the Golden Rule and other classical Judeo-Christian ethical principles.
- Conversely, whatever policies and decisions are aimed at *manipulating* customers are unethical and should be avoided, both because they are wrong and because they will surely lead to loss of market share once they are found out. There have been plenty of examples presented in this article—playing deliberate-incompatibility games, suing your second sources, and cynically misusing standards as Trojan horses.
- As CPUs become smaller and users almost universally come to understand them as components, the computing world will be better off with the semiconductor-industry-culture view that second-sourcing is a way of life than with the minicomputer-industry-culture view that it is immoral. Second-sourcing leads to

free interchange of software and add-on boxes, and thus ultimately to greater user satisfaction and more rapid improvement in technology, and even to greater national security and better service to mankind. These considerations may in time give software transportability and plug-compatibility the force of civil rights that will transcend the property rights which manufacturers hold—or think they hold—to architectures.

• A CPU architecture which has become very popular because users have found that it serves them well eventually becomes bigger than the company which originated it. Users acquire a huge investment in existing software and prefer to *retrofit* new hardware to this software rather than to rewrite it. If the economic mass of this software is sufficiently large, second sources will appear for all kinds of things which users need, including CPUs. Any attempt by the original manufacturer to stymie this process will, if successful, ultimately cause his CPU architecture to lose out in the marketplace to other architectures with more second sources.

• In consequence of the above, manufacturers should make some real effort to avoid inventing new instruction sets when they are not absolutely necessary. It *serves* the customer to give him a superior new CPU which doesn't force him to junk or rewrite all of his software. The world needs a new instruction set like it needs a new species of mosquito.

• The most durable standards seem to be those which start out with one company's definitions, spread out to cover much of the world because of high-volume production of something, become accepted in use, become somewhat diluted as slightly mutually incompatible tweaks get made in them along the way to adapt to particular user needs, and then get pulled back together by a broadly based user standardization effort. The IBM 370 instruction repertoire has by now been through most of these phases and is well on its way to becoming a world standard, even though by now IBM itself might well like to bring out some improvements which would not be IBM-370-compatible. Much the same thing can also be said of the DEC PDP-11 CPU architecture, the DG Nova architecture, and the Advanced Micro Devices 2901 bipolar microcomputer building block. In the personal computing world, the S-100 bus has already reached the very last stage of this evolution. Fill in your own favorite example—mine is the 4-foot, 8½-inch railroad track gauge, defined by the Romans two millennia before railroading began.

• *Customers don't buy hardware—they buy solutions to problems.* Even a large-magnitude improvement in hardware will get a chilly reception if it creates new problems such as incompatibility; it is worth a large amount of effort and cost to keep the new improved machine at lease upwards-compatible from the old one.

Why should manufacturers pay attention to the altruistic-sounding maxims I have stated here? Consider these words of Robert Townsend, for several years No. 1 at Avis Rent-a-Car: ". . . money, like prestige, if sought directly, is almost never gained. It must come as a byproduct of some worthwhile objective or result which is sought and achieved for its own sake."[2]

Proper thanks

There are several people I *can* publicly and gratefully thank:

Permission to use the Antonio Prohias Spy vs. Spy cartoons was kindly granted by William M. Gaines, publisher of *Mad* magazine, on behalf of the copyright owners.

The European railway gauge anecdote came from a talk by Tom Steel of System Development Corporation to the Minneapolis-St. Paul Association for Computing Machinery chapter some time during the mid-1960's. His subject was *software* standards and compatibility!

The Cincinnati streetcar and Russian artillery shell anecdotes I owe to Dick Delp, then of Signetics, who also happened to be chairman of the IEEE Computer Society Subcommittee on Floating-point standards.

There are also many other people to whom I am grateful, but whom I had better not

Epilogue

This article was originally written two-thirds of a decade ago, and time does not stand still. Some subsequent events, outcomes, comeuppances, and fiascoes deserve a wry observation or two:

- Fairchild and Data General aren't suing each other any more. Fairchild still sells 9440-series microprocessors, and DG still sells minicomputers. Neither company is quite as much of a key player in the industry as it was a few years back. Probably the real winners in this case were the law firms representing Fairchild and DG.

DG also got a judge to force Digital Computer Controls (DCC), a New Jersey start-up, to quit making a DG-compatible minicomputer, allegedly from liberated DG prints. The truly memorable aspect of this case was the judge's opinion that he would have found for DCC, rather than for DG, if DCC had done "a legitimate reverse engineering job." Subsequently, DG *bought* DCC and continues to operate it as a subsidiary.

- Two Silicon Valley congressmen, Don Edwards and Norm Mineta, got a federal law on the books in 1984 which seems to reflect much the same viewpoint as that of the New Jersey judge. Copying your competitor's silicon chip masks with a camera is a no-no, but designing your own chip from scratch to go into the same sockets isn't prohibited. So who says reinventing the wheel is always a waste of effort?
- As a group, the classical minicomputer manufacturers are losing out to hordes of newcomers whose wares revolve around high-end microprocessors. At the 16-bit level, designing minicomputers from MSI bits and pieces is a lost art, and the world is mostly split between designs using the Intel 8086 and its progeny and those using the Motorola MC68000 and its offspring, with here and there a few sockets for the Zilog Z8000 and the National NS16016. At the 32-bit level, the high-end-microprocessor people are swarming across the horizon, but they haven't yet gotten close enough to the minicomputer wagon train to shoot arrows at it. The real (i.e., silicon and not just paper) 32-bit microprocessors as of this writing seem to be the Motorola MC68020, the National NS32032, and—of all things—the NCR 32. In sum, the minicomputer companies *did* largely win the battle of putting their second sources out of business, only to lose the war to a ragtag army of little system houses with *interchangeable* boards featuring *common* microprocessors and communicating over *standard* buses. So much for locking up customers forever.
- Apple Computer, which was still literally a garage shop when I was writing the original article, has become a multimillion-dollar corporation, partly through a conscious "open architecture" strategy. This encouraged other garage shops to build enhancement boards which dropped into Apples and thereby conferred upon them such goodies as faster CPUs, CPUs that could run under other operating systems, larger memories, and so forth. In a previously undreamed-of masterstroke, Apple even let some of these little companies—and also many third-party-software houses—use their corporate logo, with the result that Apples started selling like Model T Fords. The sincerest compliment of all was paid to Apple's marketing strategy by none other than IBM, which unbuttoned enough to essentially copy it!

However, it now appears that Apple is becoming yet another example of a principle expounded by Ed Lee of Pro-Log Corporation at a recent IEEE meeting: "Once someone becomes successful, they stop being open." It has been reported that Apple made "an offer you can't refuse" to a certain little Silicon Valley industrial design house—one that has designed many Apple equipment cabinets—to stop doing similar work for Steve Wozniak, one of Apple's two founders, who has left the company to do his own thing.

- Although I truly believe that my observations about second-sourcing, incompatibility, standards committees, etc., are almost timeless, the arena of combat is shifting over time and CPU architectures and instruction sets are losing their central importance. When any third-party-software house can write its thing in Forth, C, Pascal, Modula-2, or some other language that excels at CPU-to-CPU transportability and object-code efficiency, something else—not a CPU instruction set—has to be used to lock up the pot of gold against marauding newcomers.
- The VMEbus, which started out as a creation of Motorola but was deliberately made "open," and which is now also officially championed by Signetics and Mostek, appears to be outdistancing various other buses which more clearly remain captive Trojan horses of the particular companies which first devised them.

And so the White Spy has won a few, and the Black Spy has won a few, and the battles still go on. As long as these battles don't gun down any companies close to your heart (or your paycheck!), or cause your very own personal computer to suddenly become a Stone Age relic, they're kind of fun to watch—like prime-time soaps, or even lots better. So stay tuned. —*Chuck Hastings*

publicly thank. This whole subject is at times a hot potato—people express strong opinions sharply at variance with the official positions of the organizations which pay their salaries. I have heard and appreciated many such opinions, as well as some milder ones.

So it is best that I express my appreciation in general terms, without publicly identifying all the people I am thanking!

Anyway, there are people—and even whole companies—out there sincerely trying to do the right thing. When you meet one and like their act, tell them so.

Reader Interest Survey

Indicate your interest in this article by circling the appropriate number on the Reader Interest Card.

High 159 Medium 160 Low 161

References

1. Douglas T. Ross, "Homilies for Humble Standards," *Comm. ACM,* Vol. 11, 1976, pp. 595-600.
2. Robert Townsend, *Up the Organization,* Alfred A. Knopf, New York, 1970, p. 62.

Chuck Hastings has spent the last half of the 1950's as a programmer, the 1960's as a computer architect (whatever that is), the 1970's as a hardware designer specializing in superminis, and the 1980's as a product planner in the semiconductor industry. He currently works for Monolithic Memories in Santa Clara, California. He has previously been with Data General, Microcomputer Systems Corporation (now Xebec), National Semiconductor, Itek Applied Technology, Racal-Milgo, two subsidiaries of United Telecom, Control Data, Honeywell, and TRW, and has moonlighted for various big and little companies in four states. He has a BA in physics and math from Grinnel College, an MA in math from UCLA, some additional EE and computer science courses from the University of Minnesota, and an MBA from the University of Santa Clara. He has two US patents in electrooptic mass memory technology, belongs to the ACM and IEEE, and still thinks that computers are a good way to have fun and get paid for it.

Questions about this article can be directed to Hastings at Monolithic Memories, 2175 Mission College Blvd., Santa Clara, CA 95054-1592.

Motorola's Silver Quill Program

Jackie Marsh
James J. Farrell III
Motorola, Inc.

A well planned author incentive program can turn word-shy engineers into a company's best spokespersons.

Getting engineers to write about their work for publication has always been a challenge to industrial companies. Yet, publication of such material can make a major contribution to the marketing objectives of a firm. Motorola Semiconductor Products Sector has addressed the problem by initiating an author recognition and incentive award program dubbed the Silver Quill Program.

Implemented almost 16 years ago, the Silver Quill Program encourages potential authors to publish worthy Motorola-related articles and books and to present papers at important conferences and meetings. The concept has three basic premises—peer recognition, management recognition, and a financial reward for efforts beyond normal job requirements which help achieve product marketing goals and enhance the company's image. All employees are eligible. Although most materials are of a technical nature, public interest articles related to Motorola are also encouraged.

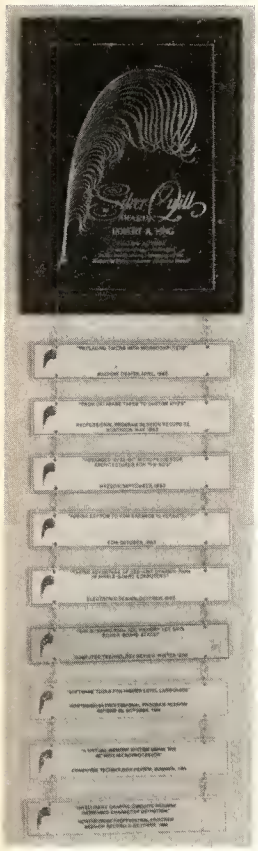
When the Silver Quill Program first began, it provided the Motorola author simply with visual recognition in the form of a plaque that could be hung on the office wall for his or her peers to see. In late 1976 however, recognizing that engineers deserve compensation for work done on their own time at home, Motorola decided to add financial incentives to the wall plaque. In addition to the monies paid by a publisher for a contributed article or book, each author is awarded an amount based on the subject matter, length of articles, quality and originality, and mode of publication. Greater credit is usually given to those articles which appear in publications or conferences with higher standards for acceptance.

The size of the awards is generally based upon the following schedule:

Articles	up to $1000
Book Chapters	up to $1500
Books	up to $5000

Not everyone receives the maximum. When the papers are individually evaluated by the review committee that oversees the awards, generally an excellent article in a top trade magazine such as *Semiconductor International, EDN, Electronic Design,* or *Electronic Products,*

0272-1732/85/0600-0053$01.00 © 1985 IEEE



Motorola employee Robert King's Silver Quill plaque shows the several awards he has already received.

or in a top IEEE publication such as *IEEE Micro, IEEE Spectrum,* or *IEEE Computer* receives about $110 per page. An excellent paper at a top conference such as IEDM, ISSCC, NCC, Wescon, or Electro is also a candidate for the top award. Articles which have appeared in one publication and are reprinted in another result in an additional award, usually, but not always, smaller than the original award.

In addition to making awards for individual publications, the Review Committee keeps track of the total value of publications awards accumulated by each individual. When an individual's aggregate value, not including books or chapters, reaches a certain value, a supplementary recognition award is made. The value of supplementary recognition is two-fold. In addition to rewarding authors for exceptional performance and achievement, it provides additional incentive for continued contributions and involvement in the program.

The path to publication

Why must a company make this kind of effort to motivate its engineers to write technical articles and papers for conferences and magazines? To understand this, you must understand the semiconductor engineer's culture. First, many engineers do not like to write to begin with. Schools exacerbate the problem by either not having (or not encouraging engineers to take) technical writing courses. Many engineers and Computer Science majors feel some apprehension writing their first article or paper. To make matters worse, the high-technology nature of the semiconductor business and the current shortage of engineers in this business keep the engineer who you are asking to take on an "extra" task extremely busy.

Defining the project. Beyond overcoming these general frustrations, engineers may need help to negotiate the long, unfamiliar path from article conception to publication. The first step in technical article generation is defining the article. An article requested by a technical publication moves through its cycle very quickly if the author produces an outline and a manuscript in a timely fashion. The other type of article submission is an outline submitted to a magazine for approval. Publishers will do one of two things—revise the outline and request a manuscript, or reject the outline. In the case of rejection, the outline may or may not be revised and sent to another magazine. Frequently, the rejecting magazine will suggest a competing or "sister" publication where publication of the proposed article would be more suitable. The essential element is the preparation of a good outline in all cases. Submitting a completed manuscript without having it previously critiqued and approved by the magazine usually guarantees failure.

Technical papers are another matter. Technical conferences generally fall into two categories—those that require submission of proposals for technical papers and those that require submission of proposals for technical sessions. In the former case, the author is usually requested to submit a 400-to-1000 word outline of his or her paper to a review committee. Frequently, a 25-to-50 word abstract is requested in addition to the outline for the purpose of prescreening when a very large number of proposals must be reviewed by the committee. Upon acceptance, the author then writes and submits his or her paper. At this stage, rejection is very unlikely. Proposing technical sessions requires far more work. First, the proposer must identify an organizer and chairperson (both are usually himself or herself in most cases, but not necessarily). Then the organizer must find four or five paper presenters for the session. Many conferences only allow one paper per "profit center" or company. This makes the job appreciably more difficult. After the organizer has found his or her five speakers, defined their papers with titles, and written a brief summary of the purpose of the session, the whole package is submitted to the conference for review. Session proposal acceptance can be very chancy. A large conference such as Wescon or Electro may have 300-to-400 sessions proposed. Less than 40 can be accepted due to time and space limitations at the conference. Other conferences with fewer proposals have a better success rate for the proposer.

Editing. The next stage in the process, interfacing an engineer to a technical editor, can be interesting. Initially, the author greets the prospect of an article rewrite with the same glee most



Pictured here are thirteen of the fourteen first-time Silver Quill recipients who accepted their awards at the March 1985 award dinner.

people reserve for an impending root canal procedure on a favorite molar. Frequently, an accepted article will be returned to an author with more revisions than the national budget. Most authors don't like this. Some tend to take these corrections, no matter how valid (and most are), very personally. Again, a lot of discussion is necessary. Many times, the editor requests a telephone interview with the author. This can be very "formal," especially if the author felt the editor did a hatchet job on his or her article. Usually, a little preliminary conference with the author can keep the interview reasonably amicable.

Finally, an aspect of technical articles that most condider almost trivial can be difficult—the biography. This would seem to fairly straightforward. It isn't. Usually all that is required is the author's educational background, a very short work history, current project, perhaps a hobby or two, a picture, and some related technical society affiliations. Now things become interesting. Some people do not like to have their picture taken. When they finally do, after several requests, they do not like any of the poses. The author ultimately selects one and it is sent off. We are not done yet. Frequently, the biography has to be done over. Some authors write several paragraphs; other include their religious beliefs or political statements. One author actually submitted his biography in the form of a limerick.

Since technical papers are usually edited internally, they normally exhibit few writing problems. For them, the major problem is presenting the paper before an audience. Some highly competent engineers and writers are weak as speakers and presenters. Although some top shows such as Electro and Wescon encourage presenters to videotape a dry run of their presentation at the show before the actual session, this may not be enough. Some presenters require three or four dry runs before they are confident enough to make a public presentation. "Confident" is the key word. Public speaking is much like bicycle riding or swimming, in that the most important thing is to overcome fear.

Approval. Before any article or paper may be published, however, it must be approved by the company on different levels. Approval is based on several factors. The first signature on the approval form, usually by the author's department head, insures that the article is technically correct and discusses a current technically sound subject. The next signature, by the operations manager, insures that the article meets the objectives of the operation. Following these, the patent and patent technology departments must sign the approval form. These signatures insure that the description is technically correct, that no unfiled patents are compromised, that no proprietary processes are revealed, and that no alternate

source agreements with other companies are violated. In addition, the patent department insures that nothing that might cause the company legal harm is published. Finally, Technical Communications signs and is responsible for sending the work to the publisher. The last signature insures that all previous levels of approval have been completed and all required revisions have been made, that the article shows Motorola in an affirmative manner, and that the article will enhance our marketing efforts.

Bringing your potential author to the "real world" of technical writing takes a good deal of work. Once they have completed their first article or paper and have seen that it wasn't as bad as they thought it would be, they generally become at least occasional authors. A few, perhaps 10 percent, became "fervent" writers. They are usually quite good. Another group, again about 10 percent, complete their article, but never write again. Finally, a few commit to write an article or paper, may or may not start it, do not complete it, and usually never attempt to write again. Generally, the majority *will* write again, if motivated by a very interesting project and encouraged by the organization.

Book publication

Although the overwhelming majority of published work is articles and papers, a few hardy souls embark upon such ambitious plans as a book or a book chapter. These authors usually fall into the "fervent" category. Book chapters are usually solicited by the primary book author who has a contract with a publisher. A typical case is when an author is responsible in his or her book for descriptions of all microprocessors currently in use. Each company manufacturing microprocessors is then solicited to supply a chapter.

Initiating and writing a book, however, takes prolonged effort. First, the concept of the book must be developed. Usually at this conceptual stage, the proposal is discussed with a technical editor at a book publishing company. If the proposal seems promising, the editor then requests an outline of the book. This outline usually runs from 15 to 30 pages in length and fully describes the intent of the book and its market, and gives a chapter-by-chapter outline of the text. The publisher then submits a copy of the outline to several paid reviewers who are very familiar with the technical topic. Invariably, the majority of the reviewers are college professors, because the publisher has hopes of getting the book "designed-in" as a standard text in a technical college course. Following this review, the author is appraised of the revisions necessary to make the book viable and acceptable to the publisher. Typically, a book will take from one to three years to produce. The company review cycle, described earlier, is accomplished on a chapter-by-chapter basis.

Support and recognition

Whether it is a book or paper that the employee has been working on at home, however, Motorola knows that spouses must support the extra effort. If the husband or wife of an author feels that he or she has made a contribution, either through active assistance, encouragement, or mere tolerance, the author has one less constraint on completing a paper. The reward of seeing hours of work culminate in the form of published material becomes a shared experience. The Publications Award Committee honors Motorola authors and their spouses (or guests) at the Silver Quill Awards Dinner. The author receives an engraved medallion to suspend from a Silver Quill wall plaque presented for the first published article or book. When the author reaches the $5,000 benchmark in accumulated awards, a Golden Quill plaque is presented. The awards presentations dinner provides a means of recognizing and sharing personal achievement with the author's peers, management, and spouse. The Silver Quill Plaque and medallions are a continual reminder of the author's contribution.

One of the most effective marketing tools is the written word. An in-depth article about a product, a process, or the organization conveys information to the biggest audience with the greatest impact. A program that instills incentive in employees to produce



Brian Spinks (center) holds his new book, *Introduction to Integrated Circuit Layout*, published by Prentice-Hall, Inc. Bernard Goodwin, (left), editor for Prentice-Hall, and Karen Gettman, Prentice-Hall representative in Austin, presented Brian with a Silver Quill and Plaque at the November, 1984 Silver Quill Awards Dinner.

educational and informative materials can only enhance the marketing efforts of the company. Contributed articles and books may appear in trade shows, seminars, schools, the news media, technical journals, and trade magazines. They are exposed to every facet of the industry, from customer to competitor. In this way, the Silver Quill Program encourages employees to participate in a marketing process with promotes recognition of, and stimulates interest in, Motorola's products, technology, and achievements.



Jackie Marsh is a technical editor for the Motorola Semiconductor Products Sector in Austin, Texas. Jackie has editorial responsibility for Memories, High-End Microprocessors, and Applications Specific, and primary supervision of Motorola's Silver Quill Program in Austin. She graduated from the University of Texas at Austin with a BBA, and held various positions before joining Motorola's Technical Communications staff in September, 1984. Her hobbies include gardening and reading science fiction.

The author's address is Technical Communications, Motorola, Inc., 3501 Ed Bluestein Blvd., Austin, TX 78721.

James J. Farrell III is the Editor-in-Chief of *IEEE Micro*. His biography appears on p. 10.

Reader Interest Survey

Indicate your interest in this article by circling the appropriate number on the Reader Interest Card.

High	162	Medium	163	Low	164

Binary-decision-based Programmable Controllers

In certain applications, BD-based controllers can react more quickly than conventional microprocessor-based controllers. Herewith additional commentary on our original discussion of these devices.

Comments on binary-decision-based programmable controllers

In a three-part article in *IEEE Micro* (August, October, and December 1983), Paul J. A. Zsombor-Murray et al. discussed binary-decision theory and described binary-decision-based programmable controllers. Although we find it encouraging to see increased interest in the area of programmable controllers based on binary-decision diagrams, we are concerned about certain claims made by Zsombor-Murray et al.—specifically the claims to being with Boute[1] the only authors that have treated the practical design of computer-like binary-decision devices for control purposes.

To the best of our knowledge, the use of binary-decision theory for the design of the control unit of a computation system can certainly be traced back to Clare.[2] Since then, much advanced research on program transformation and on implementation of algorithms by means of binary-decision-based programmable controllers has been published by American as well as by European researchers.

Lee[3] showed that binary-decision diagrams could be easily programmed and called the result (simple) decision programs; moreover, he suggested a universal instruction type that implemented the evaluation process taking place at an internal node of a binary-decision diagram. Cerny, Davio, Mange, Sanchez, and Thayse[4-7] have investigated special-purpose architecture for the execution of programs derived from binary-decision diagrams. In a recent book Davio, Deschamps, and Thayse[8] investigated and compared several microprogrammed architectures for the optimal implementation of binary-decision diagrams. Some of these architectures had already been suggested in the book by Fletcher.[9]

In the above enumeration, we have limited ourselves to the English literature; it is clear, however, that a lot of papers on the same subject have been published in the French as well in the Russian literature. So we cannot

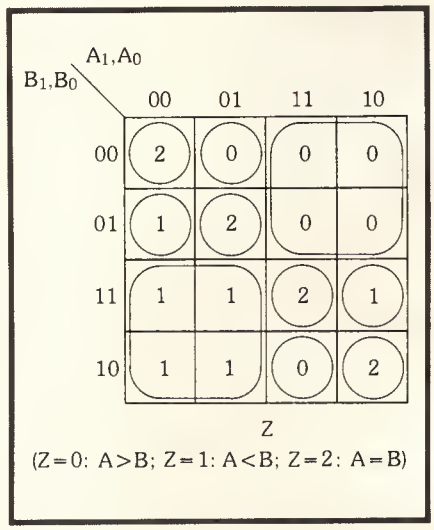


Figure 1. Partition of the Karnaugh map.

0272-1732/85/0600-0058$01.00 © 1985 IEEE

agree with another statement of Zsombor-Murray et al. that "binary-decision programming is not a widely accepted tool among digital logic designers" and that "Boute's binary-decision machine seems to be the sole exception."

Two of us (Mange and Sanchez) have used binary-decision programming and binary-decision controllers both for educational purposes[10] and for the design of industrial prototypes.[11] In the context of logic (particularly the implementation of boolean functions), another one of us (Thayse) has devised specialized methods[6-8] which attempt to use some of the standard tools of logic (such as reduction to minimal formulas) in order to build binary-decision diagrams with minimal storage. This formulation is based on a class of functions called P-functions, and the resulting method has been programmed in view of a computer treatment.[12] Besides this, a more heuristic method for obtaining a minimal binary-decision tree has been derived by Cerny and two of us (Mange and Sanchez).[4] This method makes use of the tabular representation of boolean functions by a Karnaugh map. This method has been improved by one of us (Mange)[10] for solving the problem of the two-bit magnitude comparator, which was also considered by Zsombor-Murray et al. in their article. The minimal tree is obtained from a partition of the Karnaugh map into groups of minterms which have to satisfy three properties: the set of groups covers the Karnaugh map; the groups are disjoint; and a test variable cuts the Karnaugh map into two parts without cutting any of the groups. Each group of minterms corresponds to a branch of the binary-decision tree, and the number of tests is equal to the number of groups minus one—the minimal tree is obtained from a partition into a minimal number of groups. Figure 1 illustrates the partition of the Karnaugh map leading to the minimal tree depicted in Figure 3b in Part I of the article by Zsombor-Murray et al. The minimal algorithm shown in Figure 3c of the same article can also be obtained from the Karnaugh map shown here. The symmetry of the map can be used for detecting the subprograms of the corresponding algorithms.

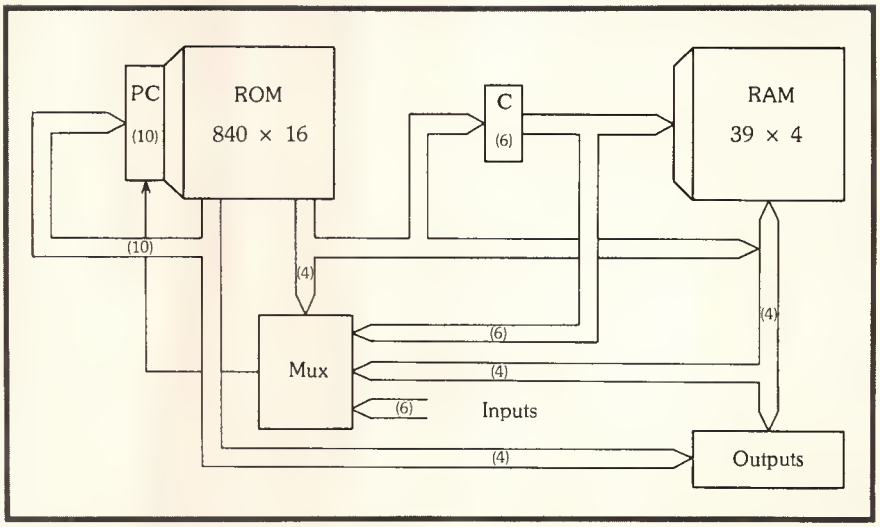


Figure 2. Architecture of an integrated controller based on a BD algorithm.

In addition to these theoretical considerations, it is worth bearing in mind that prototypes of integrated digital watches have been realized by two of us (Mange and Sanchez)[11,13] in collaboration with the Centre Electronique Horloger, an organization sponsored by the Swiss government and by the major Swiss watch manufacturers. These prototypes constitute a typical example of the application of the binary-decision algorithm to the design of integrated controllers. Because of the simplicity of the operations required by a watch, it is interesting to replace the ALU which appears in classical watch circuits with binary-decision algorithms—such replacement reduces the size of the integrated circuit. Figure 2 shows the architecture of the proposed processor; its elements include a microprogram ROM addressed by a counter PC and a data RAM (for hours, minutes, etc.) that is addressed by a register C. Figure 2 also shows the output and setup circuits, and the multiplexer which chooses the variable under test.

The test variables are the external inputs (which are the manual commands of the watch), the RAM data, and the bits of the register C. These last bits are tested for removing the stack usually used for memorizing the return address of a subprogram; we make use of binary-decision algorithms at the end of subprograms for determining the return address in terms of the data dealt with. Figure 3 illustrates the algorithm used for determining the number of days in a month; the variables M4, M3, M2, M1, and M0 give the month in BCD.

D. Mange
E. Sanchez
Swiss Federal Institute of Technology
Lausanne

A. Thayse
Philips Research Laboratory
Brussels, Belgium

References

1. R. Boute, "The Binary-Decision Machine As Programmable Controller," *Euromicro Newsletter,* Vol. 1, No. 2, 1976, pp. 16-22.
2. C. Clare, *Designing Logic Systems Using State Machines,* McGraw-Hill, New York, 1973.
3. C. Lee, "Representation of Switching Circuits by Binary Decision Programs," *Bell System Tech. J.,* Vol. 38, 1959, pp. 985-999.
4. E. Cerny, D. Mange, and E. Sanchez, "Synthesis of Minimal Binary Decision Trees," *IEEE Trans. Computers,* Vol. C-28, 1979, pp. 721-727.
5. M. Davio and A. Thayse, "Optimization of Multivalued Decision

Algorithms," *Philips J. of Research,* Vol. 33, 1978, pp. 31-65.

6. M. Davio and A. Thayse, "Implementation and Transformation of Algorithms Based on Automata," *Philips J. of Research,* Vol. 34, 1979, pp. 26-47.

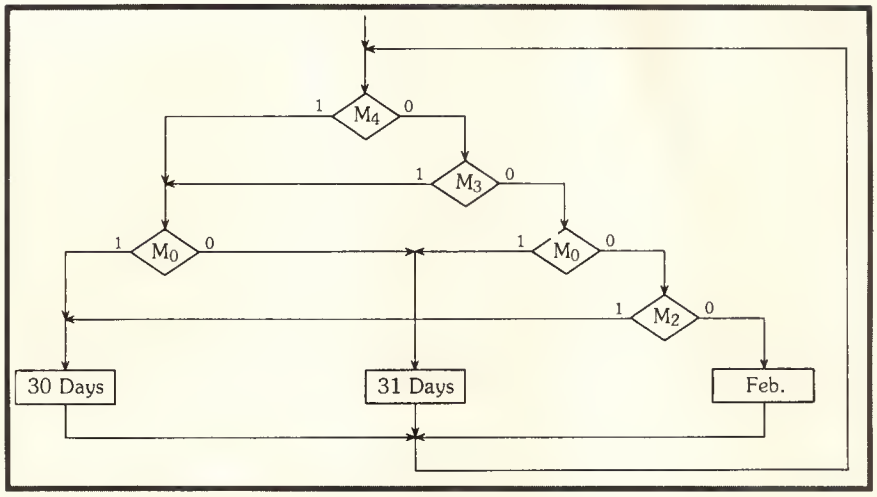


Figure 3. Algorithm for determining the number of days in a month.

7. A. Thayse, "Synthesis and Transformation of Programs by Means of P-functions," *IEEE Trans. Computers,* Vol. C-31, 1982, pp. 34-40.

8. M. Davio, J.-P. Deschamps, and A. Thayse, *Digital Systems With Algorithm Implementation,* Wiley, New York, 1983.

9. W. Fletcher, *An Engineering Approach to Digital Design,* Prentice-Hall, Englewood Cliffs, NJ, 1980.

10. D. Mange, E. Sanchez, and A. Stauffer, *Systèmes Logiques Programmés,* Presses Polytechniques Romandes, Lausanne, 1982.

11. C. Piguet, J. F. Perotto, N. Péguiron, J. J. Monbaron, E. Sanchez, and R. Saint-Girons, "Microcomputer Design for Digital Watches."

12. E. Sanchez and A. Thayse, "Optimisation and Transformation of Algorithms Based On Automata" (Part III), *Philips J. of Research,* Vol. 36, 1981, pp. 159-172.

13. E. Sanchez and A. Stauffer, "Horloge Microprogrammée," *10ème Congrès International de Chronométrie,* Genève, 1979, pp. 279-284.

Author's reply:

It is encouraging to see that BD logic and its applications are more popular than we assumed. The additional references provided by Mange, Sanchez, and Thayse are gratefully acknowledged. We should have paid more attention to the international literature. This oversight could have been easily avoided since over ten languages are spoken in our lab.

Unfortunately, others seem to be similarly guilty. There is little intersection in the Venn diagram shown in Figure 1 representing references from five sources. (A) represents the commentators' bibliography, (B) is from a recent review article on decision trees and diagrams by Moret,[1] (C) was compiled by van Driel[2] in an internal, preliminary report on BD compiler algorithms, (D) is from Mago's work[3,4] on microprocessor networks, and (E) is our bibliography.[5] The diagram comprises nearly 200 diverse publications and many research interests. The commentators are interested in logic implementation and transformation algorithms. Moret's review spans databases, machine diagnosis, testing methods, complexity measures, and other topics. Van Driel tried to optimize BD trees for combinational logic. Mago's cellular processor is pertinent to fifth-generation mainframes, according to Davis.[6] We are primarily interested in fast, simple instruments for mechanical measurement and control. Would there not be more source literature overlap if these papers dealt with boolean design? Is not our statement that "BD programming is not a widely accepted tool . . . " at least a fair, if not completely accurate, generalization? Furthermore, Moret[1] says, "Boolean functions . . . are more readily understood than general discrete functions."

It's true that we were unaware of the commentators' BD applications in education and watch design. On the other hand, we described multibit parallel output BD architecture in June 1979,[7] built a working, general-purpose BD prototype in August 1979, and presented a paper on it in December 1979.[8] Several articles and a book[9-13] imply that this feature was devised by Davio and Thayse[10] and by Fletcher[13] in 1980. As far as practical application is concerned, in 1983 Davio, Deschamps, and Thayse[14] refer only to Boute's machine[15] and to Sanchez and Stauffer's watch.[16] There are apparently no commercially available industrial controllers employing the BD principle.* Surely, the foregoing observations make it difficult to accept that BD design techniques are widely known and applied.

At the risk of semantic haggling, we disagree with the comment that our controllers are based on BD diagrams. To say so is akin to a hypothetical statement that any particular combinational logic circuit is based on Karnaugh maps. Rather, the BD diagram is but one representation of the fundamental if-then-else statement or binary switch upon which our BD controllers are based.

Our interest in BD automata grew not from any graph-theoretical consideration of BD diagrams but from an interest in developing fast, simple,

*A survey has appeared[40] listing 158 models of programmable controllers from 48 manufacturers. Not a single one is BD-based.

programmable sequence controllers.[17-19] This evolution started upon our realization that Boute's machine[15] could fulfill this role more effectively than any boolean device.

Indeed, Clare's state-qualifier-pair ROM address structure,[20] used in his algorithmic state machines, is BD-like. But this is for arithmetic computation, something that a BD machine does poorly. Neither does Clare refer to Lee[21] nor does Boute mention Clare. Although we agree that "much advanced research on program transformation and on implementation of algorithms . . . has been published" we cannot see how this bears on the design of sequence controllers.

We must also contradict the commentators' interpretation of Lee's contribution.[21] He writes only of BD programs. Nowhere does he suggest that his universal instruction occupies a BD tree node. Akers[22] probably deserves credit for introducing the idea of BD diagrams and trees.

As we stated in our article, BD program minimization is roughly at the Karnaugh map stage. The map given as Figure 1 of the commentators' letter (and originally published as Figure 3 in Mange[23] does generate the trellis for a two-bit magnitude comparator, but we should note the following:

- Karnaugh maps help to minimize combinatorial functions with one-bit outputs, but BD logic is often used to implement multiple-output and even sequential logic.
- Functions with more than six inputs are not uncommon, but their maps are confusing and not very useful. One seldom uses even five-input maps.

There is evidence[24] that the optimization of decision trees with polynomial-size inputs is an NP-hard problem. Therefore, we must be satisfied with facilities which quickly develop nearly optimal trees. Moret[1] says "Synthesis of optimal decision trees by P-functions requires exponential time and is therefore of little practical interest." Hudson's pattern matching algorithm[25,26] is probably suboptimal; however, it has been found to be quite useful because

- we have not been able, by manual methods, to improve upon the highly pruned and trellised trees which it produces, and
- it is very fast.

On the theoretical side, van Driel[2] made a preliminary investigation in 1982 of optimization methods. He made some progress in two ways:

- *Decision table compression.* The Quine-McCluskey method[27] was used to reduce the initial decision or truth table by eliminating redundant output states. Shwayder[28] suggested this procedure, although it is not optimal, because it is well structured and therefore easily programmed.
- *Tree minimization.* Here, there are two somewhat conflicting objectives: minimization of tree nodes to produce the smallest program, or minimization of average path length and hence of program running time.

The minimum node algorithm was adapted from Pollack,[29] and in a test it automatically reproduced the trellis derived manually by Le-Ngoc.[7] The minimum path length algorithm is from Shwayder[30] and is based upon Huffman coding.[31] It selects from the compressed decision table the row with the highest entropy in one step and is therefore suboptimal because it optimizes each node independently. We started work on finding a compromise between examining single nodes and examining all possible trees. Since Huffman coding satisfies

$$H(Y) \leqslant \bar{n} \leqslant H(Y) + 1$$

where $\bar{n}$ is the average path length, H(Y) is the entropy function $\Sigma_i - Pi \bullet \log_2 Pi$, and Pi is the probability of output, P, a "near" optimal solution can be determined by calculating the Huffman solution and $1 - H(Y)$ as a first criterion. The difference between $\bar{n} - H(Y)$ from any proposed algorithm and from the Huffman solution is a test of "goodness."

We suspended this work temporarily because the pattern matching algorithm incorporated in the μP/BD hybrid machine operating system[32,33] is in service and is satisfactory. We are presently occupied with trying to find some systematic way to treat sequential logic with BD programming, perhaps via a dynamic programming approach.[34]

We might add to the commentators' short list of educational and practical applications of BD by citing Kucuk's balancer,[35] Levi's weigh-scale,[36] the dynamic transducer,[37] and the binary sensor matrices with variable reference,[38] all developed in the DATAC Lab. Our students are exposed to BD programming and architecture at the undergraduate and graduate levels. Notwithstanding that, it is doubtful that many digital designers in Switzerland and Canada, let alone the rest of the world, have ever heard of, much less regularly used, BD techniques!

We hope this response has allayed the concerns of the commentators regarding our claim to having contributed to BD application. These contributions can be summarized as

- recognition of the need for, and incorporation of, multibit, parallel BD output,
- definition of particular applications where BD architecture is an attractive alternative to hardwired boolean logic and to conventional microprocessors, and
- the writing of articles to get engineers interested in BD techniques.

It is often hard to tread the thin line between redundancy with, and acknowledgment of, others' work.

We note two oversights in the commentators' bibliography. Reference 4

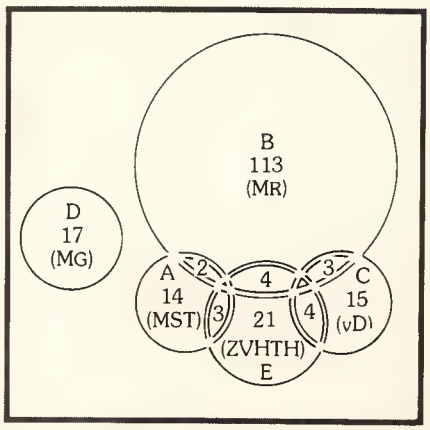


Figure 1. Venn diagram of the literature.

(Cerny, Mange, and Sanchez) did appear in the *IEEE Transactions on Computers,* Volume C-28, but on pages 472-482. We presume that Reference 6 was intended to be M. Davio and A. Thayse, "Implementation and Transformation of Algorithms Based on Automata, Part I," *Philips Journal of Research,* Volume 35, pages 122-144. An article by the same authors, "Algorithms for Minimal-Length Schedules," does appear in Volume 34, pages 26-47, but it deals with a formal critical-path approach to optimizing minimal multitask systems and, in our opinion, it hasn't much to do with BD programming.

Our paper was initially submitted to *IEEE Micro* in the spring of 1982. It is not surprising that we were unaware of the work by Mange, Sanchez, and Stouffer[39] (from 1982) and of the work by Davio, Deschamps, and Thayse[14] (from 1983).

We are pleased that our article has attracted comments from Mange, Sanchez, and Thayse, researchers whose reputation commands the highest admiration. Debates such as this will inevitably lead to a better mutual understanding of the largely complementary goals of, and results achieved by, theoreticians and designers. We hope the use of BD architectures and programming in a variety of applications will be promoted thereby.

P. J. Zsombor-Murray
L. J. Vroomen
R. Hudson
T. Le-Ngoc
DATAC Computer Laboratory
McGill University
Montreal

References

1. B. M. Moret, "Decision Trees and Diagrams," *Computing Surveys,* Vol. 14, No. 4, Dec. 1982, pp. 593-623.
2. C.-J. L. van Driel, "Binary Decision Compiler Algorithms," Mech. Eng. Research Labs, Memo 82-2, McGill University, Montreal, Sept. 1982.
3. G. A. Mago, "A Network of Microprocessors to Execute Reduction Languages, Part I," *Int'l J. of Computer and Information Sciences,* Vol. 8, No. 5, 1979, pp. 349-385.
4. G. A. Mago, "A Network of Microprocessors to Execute Reduction Languages, Part II," *Int'l J. of Computer and Information Sciences,* Vol. 8, No. 6, 1979, pp. 435-471.
5. P. J. Zsombor-Murray, L. J. Vroomen, R. Hudson, T. Le-Ngoc, and P. Holck, "Binary-Decision-Based Programmable Controllers—Part 1," *IEEE Micro,* Vol. 3, No. 4, Aug. 1983, pp. 67-83.
6. A. L. Davis, "Computer Architectures," *IEEE Spectrum,* Vol. 20, No. 11, Nov. 1983, pp. 94-99.
7. T. Le-Ngoc, P. J. Zsombor-Murray, and L. J. Vroomen, "A Binary-Decision Approach to Industrial Programmable Controllers," *Proc. Int'l Symp. Measurement and Control,* Grenoble, June 1979, pp. 57-61.
8. P. J. Zsombor-Murray, L. J. Vroomen, T. Le-Ngoc, and P. Holck, "A Binary-Decision-Based Programmable Controller," *Proc. Second Int'l Symp. on Mini and Microcomputers in Control,* Fort Lauderdale, FL, Dec. 1979, pp. 60-65.
9. M. Davio and A. Thayse, "Optimization of Multi-valued Decision Algorithms," *Philips J. of Research,* Vol. 33, No. 1/2, 1978, pp. 31-65.
10. M. Davio and A. Thayse, "Implementation and Transformation of Algorithms Based on Automata, Part I," *Philips J. of Research,* Vol. 35, No. 2, 1980, pp. 122-144.
11. A. Thayse, "Implementation and Transformation of Algorithms Based on Automata, Part II," *Philips J. of Research,* Vol. 35, No. 3, 1980, pp. 190-216.
12. E. Sanchez and A. Thayse, "Implementation and Transformation of Algorithms Based on Automata, Part III," *Philips J. of Research,* Vol. 36, No. 3, 1981, pp. 159-172.
13. W. Fletcher, *An Engineering Approach to Digital Design,* Prentice-Hall, Englewood Cliffs, NJ, 1980.
14. M. Davio, J. P. Deschamps, and A. Thayse, *Digital Systems with Algorithm Implementation,* Wiley, New York, 1983.
15. R. T. Boute, "The Binary Decision Machine as Programmable Controller," *Euromicro Newsletter,* Vol. 1, No. 2, 1976, pp. 16-22.
16. E. Sanchez and A. Stauffer, "Horloge Microprogrammée," *10ème Congrès Int'l de Chronométrie,* Genève, 1979, pp. 279-284.
17. T. Le-Ngoc, P. J. Zsombor-Murray, and L. J. Vroomen, "Evolution of Data Acquisition and Control Terminals in Distributed Systems," *Proc. First Int'l Symp. on Mini and Micro Computers in Control,* San Diego, CA, Jan. 1979, pp. 30-36.
18. R. L. Tabachnick, T. Le-Ngoc, L. J. Vroomen, and P. J. Zsombor-Murray, "Automated Design of Industrial Sequential Controllers," *Proc. Sixth Man-Computer Comunications Conf.,* Ottawa, Ont., May 1979, pp. 99-113.
19. R. L. Tabachnick, P. J. Zsombor-Murray, L. J. Vroomen, and T. Le-Ngoc, "Sequence Controllers with Standard Hardware and Custom Firmware," *IEEE Micro,* Vol. 1, No. 2, May 1981, pp. 9-25.
20. C. Clare, *Designing Logic Systems Using State Machines,* McGraw-Hill, New York, 1973.
21. C. Y. Lee, "Representation of Switching Circuits by Binary-Decision Programs," *Bell System Tech. J.,* July 1959, pp. 985-999.
22. S. B. Akers, "Binary Decision Diagrams," *IEEE Trans. Computers,* Vol. C-27, No. 6, June 1978, p. 509-516.
23. D. Mange, "Arbres de Decision pour Systemes Logique Cables ou Programmes," *Tirage a part du Bulletin ASE/UCG,* (Association Suisse des Electriciens), Vol. 69, No. 22, 1978, pp. 1238-1243.
24. B. M. Moret, M. G. Thomason, and R. C. Gonzalez, "The Activity of a Variable and Its Relation to Decision Trees," *ACM Trans. Programming Languages and Systems,* Vol. 2, No. 4, Oct. 1980, pp. 580-595.
25. R. Hudson, P. J. Zsombor-Murray, and L. J. Vroomen, "Operating System for a Hybrid BD/μP Programmable Controller," *Proc. Twentieth ISMM Int'l Symp. on Mini and Micro Computers,* Cambridge, MA, July 1982, pp. 1-4.
26. R. Hudson, "The Design of a Hybrid Microcomputer/BD Machine Industrial Controller," M.Eng. thesis, McGill University, Montreal, in preparation.
27. E. Mendelson, *Boolean Algebra and Switching Circuits,* McGraw-Hill, New York, 1970.
28. K. Shwayder, "Combining Decision Rules in a Decision Table," *Comm. ACM,* Vol. 18, No. 8, Aug. 1975, pp. 476-480.
29. S. L. Pollack, "Conversion of Limited-Entry Decision Tables to Computer Programs," *Comm. ACM,* Vol. 8, No. 11, Nov. 1965, pp. 677-682.
30. K. Shwayder, "Extending the Information Theory Approach to Converting Limited-Entry Decision Tables to Com-

puter Programs," *Comm. ACM*, Vol. 17, No. 9, Sept. 1974, pp. 532-537.

31. D. A. Huffman, "A Method for the Construction of Minimum Redundancy Codes," *Proc. IRE*, Vol. 40, No. 10, Oct. 1952, pp. 1098-1101.
32. R. Hudson, L. J. Vroomen, P. J. Zsombor-Murray, and T. Le-Ngoc, "A Hybrid Binary-Decision/Microprocessor Programmable Controller," *Proc. 4th IASTED Int'l Symp.*, Cairo, Sept. 1981, pp. 109-114.
33. P. J. Zsombor-Murray, L. J. Vroomen, R. Hudson, T. Le-Ngoc, and P. Holck, "Binary-Decision-Based Programmable Controllers—Part 3," *IEEE Micro*, Vol. 3, No. 6, Dec. 1983, pp. 24-39.
34. A. Martelli and U. Montanari, "Optimizing Decision Trees Through Heuristically Guided Search," *Comm. ACM*, Vol. 21, No. 12, Dec. 1978, pp. 1025-1039.
35. A. C. Kucuk, P. J. Zsombor-Murray, and L. J. Vroomen, "A Binary-Decision-Controlled Robotic Balancer," *Proc. CAN/CAM Conf. Applied Mechanics*, Saskatoon, Sask., May 1983, pp. 855-856.
36. M. H. Levi, L. J. Vroomen, and P. J. Zsombor-Murray, "Computerized Batch Weighing Systems Using Binary-Decision-Based Intelligent Reflexive Interface Modules," *Proc. Mini and Micro Computers and Their Applications Conf.*, Lugano, Switzerland, June 1983, pp. 83-87.
37. P. J. Zsombor-Murray, L. J. Vroomen, A. C. Kucuk, and R. Hudson, "BD-Based Displacement, Velocity, and Acceleration Transducer," *Proc. Measurement and Control '83*, Athens, Greece, Aug. 1983.
38. M. H. Levi, L. J. Vroomen, and P. J. Zsombor-Murray, "IRI-Based Multi-Processor System for the Evaluation of a Comparator Matrix with Variable Reference," *Proc. Modeling Identification and Control Conf.*, Feb. 1984, Innsbruck, Austria.
39. D. Mange, E. Sanchez, and A. Stauffer, *Systèmes Logiques Programmés*, Presses Polytechniques Romandes, Lausanne, 1982.
40. W. R. Flynn, "Range of PC Offerings Continues to Grow," *Control Engineering*, Vol. 31, No. 1, Jan. 1984, pp. 82-86.

In the response above, Paul J. Zsombor-Murray and his colleagues acknowledged contributions to BD theory and practice made by Mange, Sanchez, Thayse, and others. Below, Zsombor-Murray and his colleagues comment on the BD-based-control application presented by Mange et al. —*Ed.*

Logical choice and the discriminating designer

Logic designers and programmers sometimes earn their keep by choosing efficient implementations of digital functions. Their options include gate logic, programmable *n*-state automata, and microprocessors programmed with table-look-up (TLU) and/or logical procedures. These are often mixed to produce a good design. Here, as a response to some issues raised by Mange, Sanchez, and Thayse, we examine and compare the different ways to design the part of a digital watch that decides which months have 31, 30, 29, or 28 days—the last-day-of-the-month discriminator.

The discriminator. Even simple digital watches can display the date as a month and day of the Gregorian calendar. Seven months have 31 days, four have 30, and, ignoring the once-a-century non-leap-year, February has 28 (and 29 in years divisible by four). Some logic is required to reset the day counter and to increment the month counter after every last day of the month.

We've never attempted a complete digital watch design, but the month discriminator problem provides an interesting exercise because it's non-trivial and yet simple enough to be used to illustrate a number of different design approaches. Mange, Sanchez, and Thayse aroused our curiosity when they stated that a Swiss watch with a binary-decision (BD) discriminator algorithm had been produced. They presented a decision tree which examined four bits, M0, M2, M3, and M4, to derive one of three outputs designated "31," "30," or "February," as shown in Figure 1. These bits, together with M1, which is not used by the discriminator, represent the BCD-coded month sequence numbers, 1 to 12. The tree contains five decision nodes; the minimum path length is two, and the maximum is four. It has a frequency-weighted average length of about 2.67. One might ask:

- Is this a good digital design?
- Is this application of BD logic, compared to gate logic, advantageous?
- Can the BD program be improved?

Peatman[1] says that the specification of a good digital system structure gives far greater opportunity for cost and complexity reduction than do formal techniques. For automata with more than four states, good state assignment is a matter of luck, insight, and experience.

In this vein, Mange, Sanchez, and Thayse claim that their use of BD logic " . . . reduces the size of the integrated circuit," but had straight binary, rather than 8-4-2-1 BCD, counters been used, the number of data variables would have been reduced by 15 percent from 39 to 33, as shown in Table 1. A binary counter circuit is also slightly simpler than its BCD counterpart.

BCD architecture was probably chosen so as to conveniently accom-

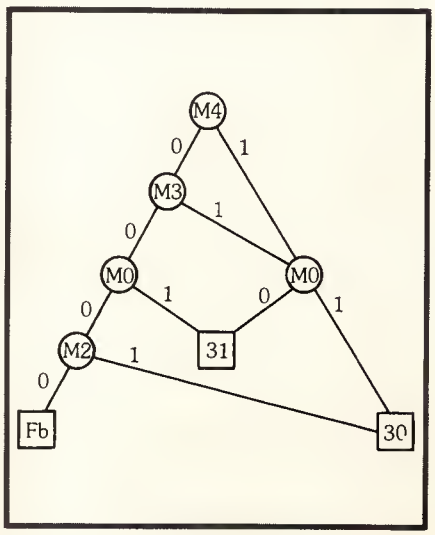


Figure 1. BCD discriminator BD tree.

Table 1.
Digital watch parameters—BCD vs. binary.

Param.	Count range	BCD	Binary	Count range
Years	00-99	Y7 Y6 Y5 Y4 Y3 Y2 Y1 Y0	Y6 Y5 Y4 Y3 Y2 Y1 Y0	$00-$63
Months	1-12	M4 M3 M2 M1 M0	M3 M2 M1 M0	$1-$C
Days	1-31	D5 D4 D3 D2 D1 D0	D4 D3 D2 D1 D0	$1-$1F
Hours	00-23	H5 H4 H3 H2 H1 H0	H4 H3 H2 H1 H0	$0-$17
Minutes	00-59	U6 U5 U4 U3 U2 U1 U0	U5 U4 U3 U2 U1 U0	$0-$3B
Seconds	00-59	S6 S5 S4 S3 S2 S1 S0	S5 S4 S3 S2 S1 S0	$0-$3B

Table 2.
Two-digit display decoder functions—BCD vs. binary.

Count	BCD	Binary	Display segment 1	Display segment 0
	M4 M3 M2 M1 M0	M3 M2 M1 M0	a1 b1 c1 d1 e1 f1 g1	a0 b0 c0 d0 e0 f0 g0
1	0 0 0 0 1	0 0 0 1	0 0 0 0 0 0 0	0 1 1 0 0 0 0
2	0 0 0 1 0	0 0 1 0	0 0 0 0 0 0 0	1 1 0 1 1 0 1
3	0 0 0 1 1	0 0 1 1	0 0 0 0 0 0 0	1 1 1 1 0 0 1
4	0 0 1 0 0	0 1 0 0	0 0 0 0 0 0 0	0 1 1 0 0 1 1
5	0 0 1 0 1	0 1 0 1	0 0 0 0 0 0 0	1 0 1 1 0 1 1
6	0 0 1 1 0	0 1 1 0	0 0 0 0 0 0 0	1 0 1 1 1 1 1
7	0 0 1 1 1	0 1 1 1	0 0 0 0 0 0 0	1 1 1 0 0 0 0
8	0 1 0 0 0	1 0 0 0	0 0 0 0 0 0 0	1 1 1 1 1 1 1
9	0 1 0 0 1	1 0 0 1	0 0 0 0 0 0 0	1 1 1 1 0 1 1
10	1 0 0 0 0	1 0 1 0	0 1 1 0 0 0 0	1 1 1 1 1 1 0
11	1 0 0 0 1	1 0 1 1	0 1 1 0 0 0 0	0 1 1 0 0 0 0
12	1 0 0 1 0	1 1 0 0	0 1 1 0 0 0 0	1 1 0 1 1 0 1

Table 3.
Counter/seven-segment display decoding rules.

	BCD	Binary
Least significant digit		
(a0)	$\overline{M0 \oplus M2} + M1 \oplus M3$	$\overline{M0 \oplus M2} + M1 \oplus M3$
(b0)	$\overline{M2} + \overline{M0 \oplus M1}$	$\overline{M2} + \overline{M0 \oplus M1}$
(c0)	$M0 \cdot \overline{M1} \cdot M2$	$M2 \oplus M3 + M0$
(d0)	$\overline{M0 \oplus M1 \oplus M2 \oplus M3} + \overline{M0} \cdot \overline{M2}$	$\overline{M0 \oplus M1 \oplus M2 \oplus M3} + \overline{M0} \cdot \overline{M2}$
(e0)	$\overline{M0 + \overline{M1} \cdot M2}$	$\overline{M0 + \overline{M1} \cdot \overline{M3}}$
(f0)	$M3 + \overline{M0} \cdot \overline{M1} + \overline{M0} \cdot M2 + \overline{M1} \cdot M2$	$\overline{M2 \oplus M3} + M0 \cdot M1$
(g0)	$M1 \oplus M2 \oplus M3 + \overline{M0} \cdot M2$	$M1 \oplus M2 \oplus M3 + \overline{M0} \cdot M2$
Most significant digit		
	(a1) = (d1) = (e1) = (f1) = (g1) = 0	(a1) = (d1) = (e1) = (f1) = (g1) = 0
	(b1) = (c1) = M4	(b1) = (c1) = M3 • (M2 + M1)

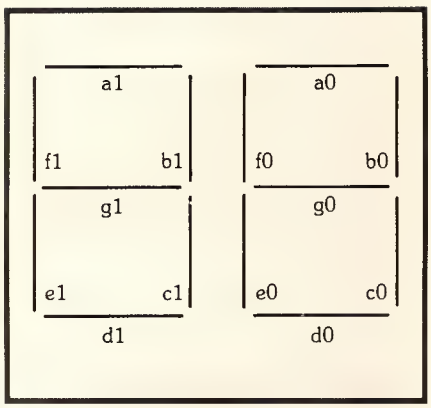


Figure 2. Display segment variables.

modate seven-segment decimal displays. Decoding is still necessary, however. Because all time components are confined to two digits, i.e., Y, M, D, H, U, and S, and all appear as two figures on the watch face, it is just as easy to decode a binary count. Since the watch contains a special VLSI chip, the existence of conventional MSI BCD-to-seven-segment decoder/drivers is irrelevant. Figure 2 shows the segments of a two-digit display. Table 2 relates the states of the two types of counter to the display states. Neither counter is standard. The BCD counter appears to be full, with M4 setting normally after "9," but it clears after "2" in the second decade and skips back to "1," not "0." The binary counter skips "0," "$D," "$E," and "$F."

The boolean equations listed in Table 3 were derived for each of 14 segments of a two-digit display for both counter types. Notice how, for the least significant digit, the BCD and the binary counts are decoded by the same expression in four out of seven cases. Figure 3 shows the Karnaugh map for segment c0.

Having established that a binary counter doesn't appreciably complicate the display interface logic, let's now proceed to examine the discriminator logic described by the BD diagram in Figure 1. Note that it is incomplete; it doesn't resolve leap years. This deficiency will be overcome shortly. Let's then present, as a comparison to the BD logic discussed by Mange, Sanchez, and Thayse, gate implementations of both BCD and binary-based

discriminators. Call the discriminator's output D*, as defined in Table 4.

The incomplete device omits D* = 01, the second line in Table 4, and uses only M-inputs. Y-inputs are necessary to complete the discriminator so that it can identify years divisible by four. Figure 4 shows the Karnaugh maps for D*, derived from the five M-inputs of a BCD counter. Though a five-input map may seem formidable, these inputs result in a pair of quite simple boolean functions. Its equivalent gate circuit can be assembled from the three gates shown in the shaded area in Figure 6a. The advantage gained by using a binary M-counter is once more evident. As can be seen in Figure 5, the boolean logic is simplified, and the shaded area in Figure 7 shows how this incomplete discriminator can be formed with just two gates.

Figure 6a shows the three additional gates required to complete the BCD discriminator; however, as shown in Figure 6b, a minimum circuit can be formed with only five gates. On the other hand, the binary discriminator can be minimally completed with two gates and two Y-inputs. This simplification is possible because binary divisibility-by-four is easily recognized as two least significant zeroes. Conversely, one must establish that the decade digit of a BCD Y-counter is odd before checking if (Y1,Y0) = 10 instead of 00. The complete binary discriminator is shown in Figure 7.

Figure 8 shows the complete, leap-year discriminating BD tree for BCD coding. Nine decision nodes are required and the worst case "February" output is achieved in seven steps. Figure 9 shows the considerably simplified tree for binary-coded months and years. Only six nodes are required—"31" is always reached in two steps; "February-leap year" in five. Tables 5 and 6 show the BD machine program code for the two complete discriminators. The program for binary Y- and M-counters can be written with ten BD instructions but the other one takes thirteen.

Programming example. We recently had to write an MC6809 assembler language subroutine to calculate the day of the year as a number from 1 to 365 (or 366) using information stored

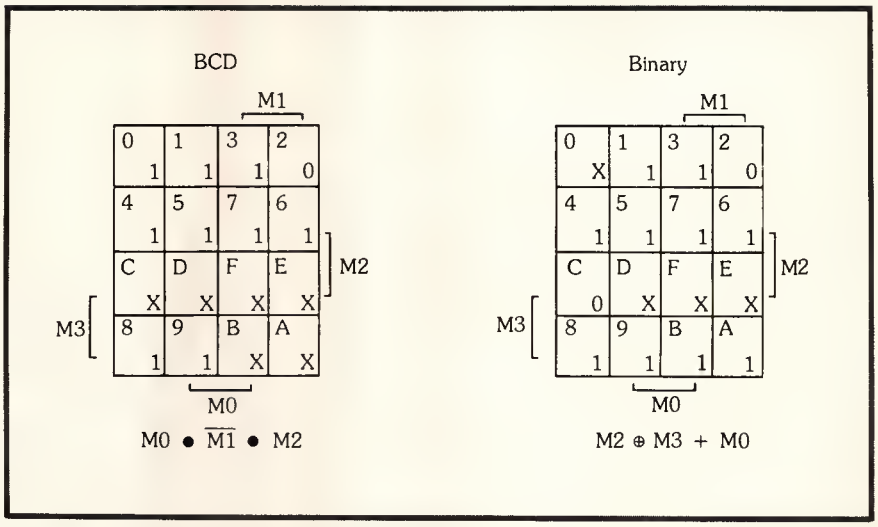


Figure 3. Least significant digit segment, c0.

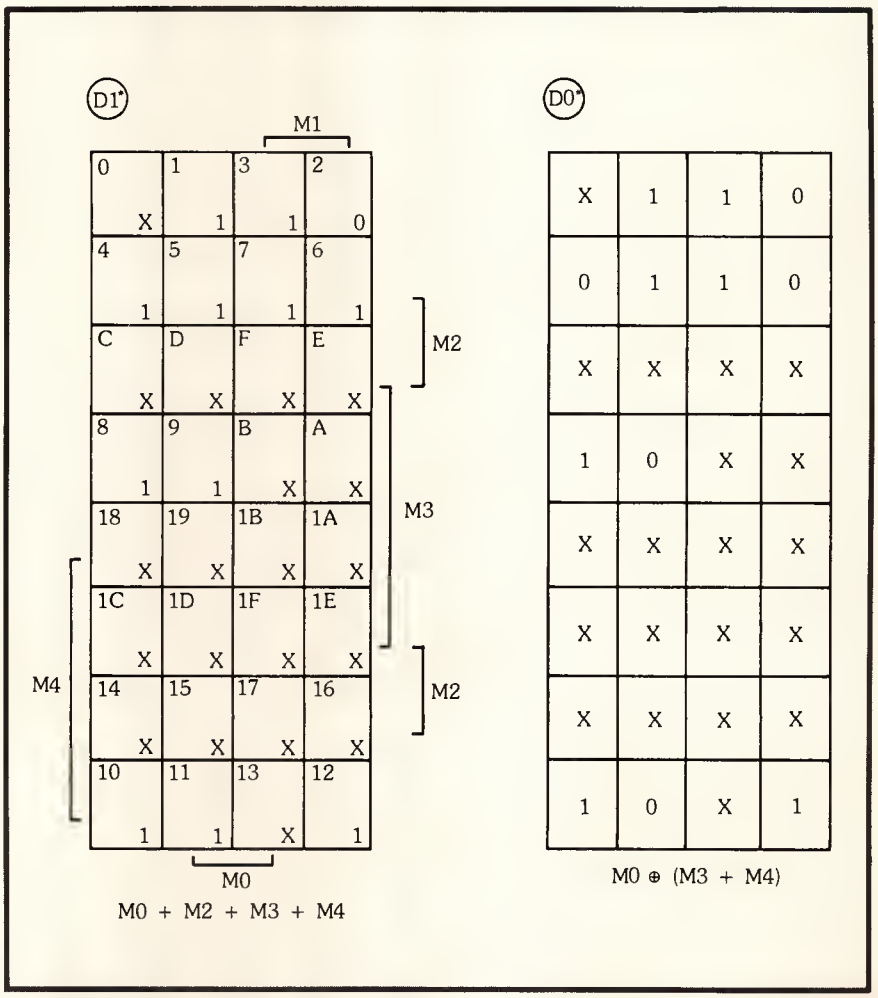


Figure 4. BCD discriminator map.

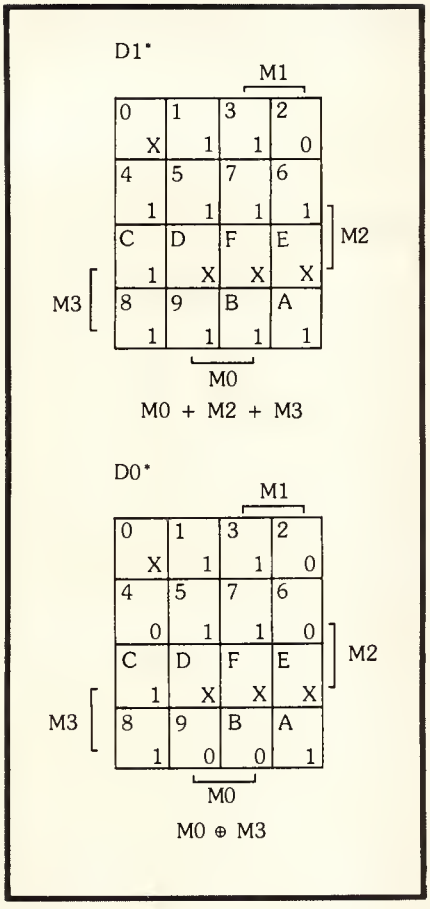


Figure 5. Binary discriminator map.

as three 8-bit integers representing month, day of the month, and year. This opportunity allowed us to write and compare three different month discriminator programs:

- Boolean—this routine requires 68 bytes and takes 65 cycles to establish "December," the maximum, and 25 cycles for "January," the minimum. The four-year average is 47.5 cycles.
- Table look-up (TLU)—this routine uses 37 bytes and calculates "February-leap year" in 48 cycles and all other cases in 37 cycles. The average = 37.7 over four years.
- Programmed BD logic—this routine needs 47 bytes and gets "February-leap year" in 46 cycles, maximum, and nine of the twelve cases in 34, minimum. The four-year average is 35.4. But if the date is stored in Canadian format, i.e., day, month,

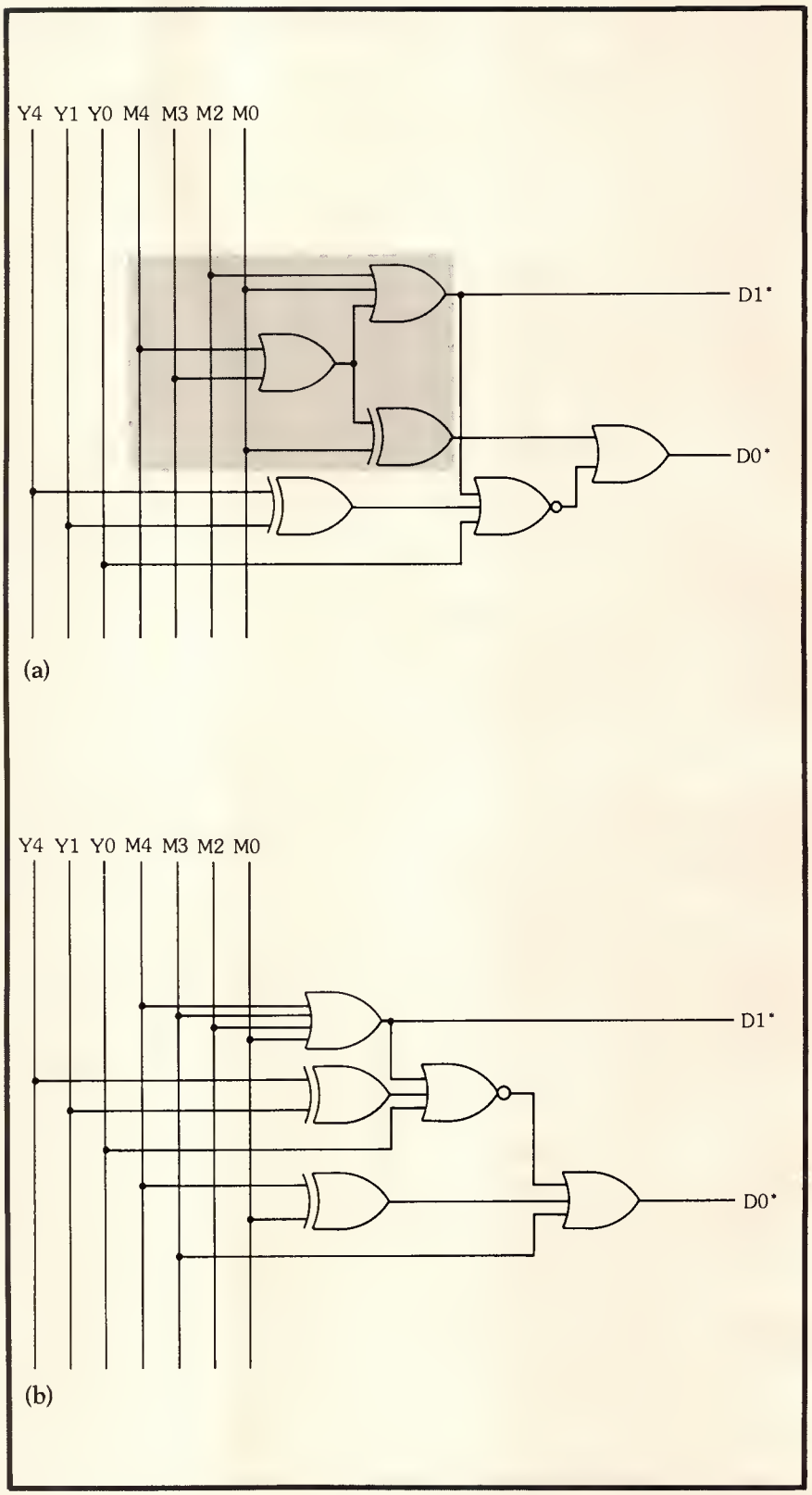


Figure 6. BCD discriminator circuit (a); BCD discriminator circuit, minimized (b).

year, this average diminishes to 31.4 cycles.

Admittedly, this won't start a rage in BD-style microprocessor programming or even save vast quantities of mainframe computing time. Nevertheless, we have here a concerete example showing that logic implemented as software and using a BD model as a programming template can run even faster than TLU.

Conclusions. This exercise in comparative logic design led us to conclude, in answer to the three questions

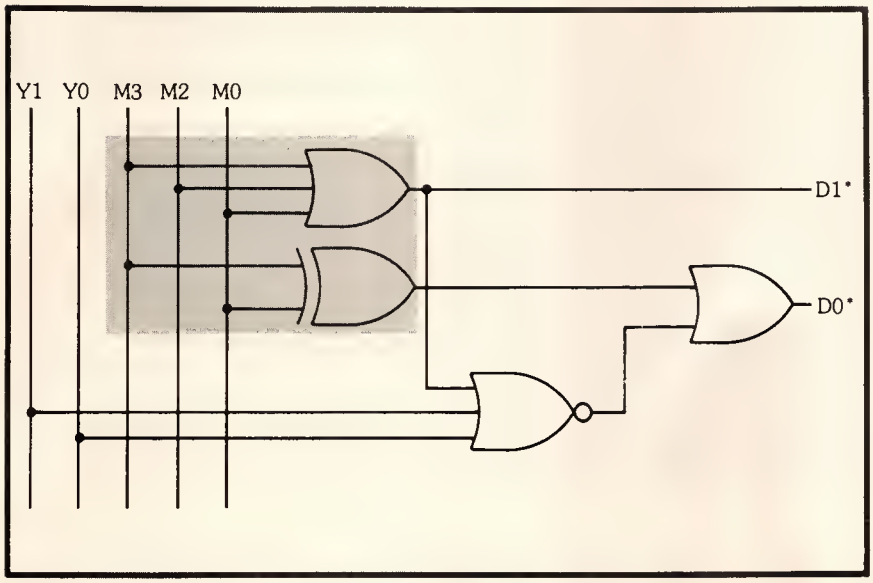


Figure 7. Binary discriminator circuit.

Table 4.
The discriminator's output.

Maximum days	Months	D1*	D0*
28	February, non-leap year	0	0
29	February, leap year	0	1
30	Apr., Jun., Sep., Nov.	1	0
31	Jan., Mar., May, Jul., Aug., Oct., Dec.	1	1

Table 5.
Binary-decision program for a BCD discriminator.

		Branch			
Line number	Input and test	True	False	Unconditional	Output
0	M4		4		
1	M0		3		
2				0	(30)
3				0	(31)
4	M3	1			
5	M2	2			
6	M0	2			
7	Y0	A			
8	Y1	B			
9	Y4	C			
A				0	(28)
B	Y4	A			
C				0	(29)

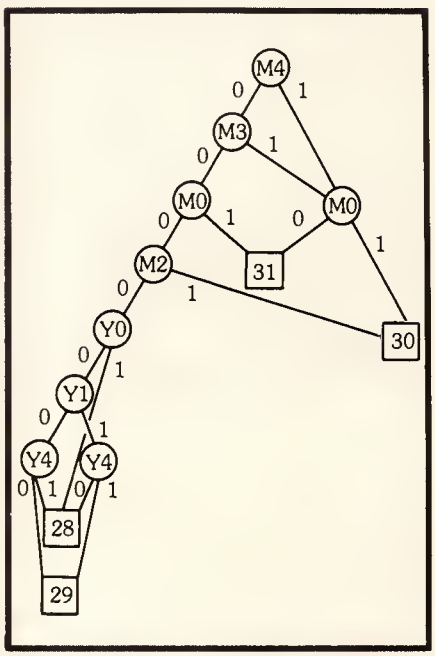


Figure 8. Complete BCD discriminator BD tree.

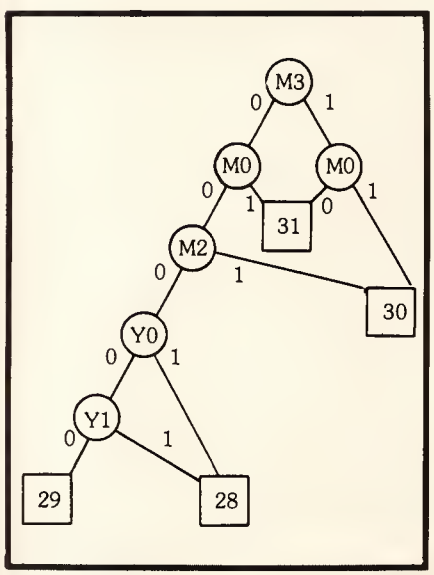


Figure 9. Complete binary discriminator BD tree.

we posed at the beginning of this response, that

- the design of a last-day-of-the-month discriminator can be improved substantially by substituting straight binary counters for 8-4-2-1 BCD month-and-year counters,
- a gate logic discriminator would occupy far less chip real estate than the equivalent BD program in ROM,
- binary counters also simplify a BD discriminator, and, furthermore,
- BD programming techniques can accelerate the execution of conventional programs. As BD program optimization tools are developed and improved, these should be incorporated into high-level-language compilers so as to help generate more efficient machine code.

Table 6.
Binary-decision program for a binary discriminator.

		Branch			
Line number	Input and test	True	False	Unconditional	Output
0	M3		4		
1	M0		3		
2				0	(30)
3				0	(31)
4	M0	3			
5	M2	2			
6	Y0		8		
7				0	(28)
8	Y1	7			
9				0	(29)

P. J. Zsombor-Murray
L. J. Vroomen
DATAC Computer Laboratory
McGill University
Montreal

Reference

1. J. B. Peatman, *The Design of Digital Systems,* McGraw-Hill, New York, 1972, pp. 32 and 47.

Editor:

I found the three-part article "Binary-Decision-Based Programmable Controllers" (August, October, and December 1983) both interesting and informative. However, by presenting just one type of finite-state machine solution (the binary-decision controller), the authors have not done justice to the vast field of potential controller solutions. In many applications, a 2^2- or 2^3-decision-based controller would be far superior to a 2^1-decision (binary decision) controller, and would not require any additional design effort or hardware. In other applications, a state machine with just a few states would make the best controller. There is much potential power and flexibility to be gained by considering controller problems from the standpoint of a finite-state machine.

This is not to say that BD controllers do not have their place, but that modern state machine design gives the engineer many potential designs to choose from, only one of which is a BD controller. The concepts used in Boute's machine are standard tools of the state machine designer. They are used to some degree in all ROM-controlled state machines. Microprogrammed state machines are a good example of 2^n-decision machines. In other words, the designer should not be restricted to a binary-input state machine, but should select the best machine for the job.

To demonstrate the validity of a generalized state machine approach, I refer to the design exercise outline in Part III of Dr. Zsombor-Murray's article. Contrary to statements made in that article, I believe the optimal solution to the exercise is a simple 12-state 2-input (plus a reset input) state machine. The designs that follow are simple, inexpensive, not hardwired, easy to design, and—if one substituted static RAM and a little interface logic for the PROM—programmable. The flexibility of state machine design also allows custom tailoring for the benefit of specific implementations. Some possible variations include toggling or nontoggling outputs as well as the type of hardware used for the input-forming logic.

Figure 1a is the state diagram for the biphase encoder given in Part III (Figure 13a) of the article. By making two simple modifications, the state diagram of this encoder can be transformed to the one in Figure 1b. The specific modifications are: (1) allowing the state transitions to take place at an arbitrary time (not on an input transition), and (2) adding one

Table 1.
Performance of the BD machine vs. state machine A (Figure 1b) and state machine B (Figure 3).

Function	Time (μs)		
	BD machine	State machine A	State machine B
Setup	2	1	1
No count	2	1	1
Count up	3	2	1
Count down	3	2	1

buffer state on each transition path. The state machine graph of Figure 1b can be implemented very easily as the ROM-controlled state machine (state machine A) shown in Figure 2. This is a two-chip (20-pin TTL), entirely soft-wired implementation. It outperforms by 50 to 100 percent the BD controller in the benchmarks (see Table 1).

However, this is not an optimal design. It was chosen only to show how easily an equivalent basic state

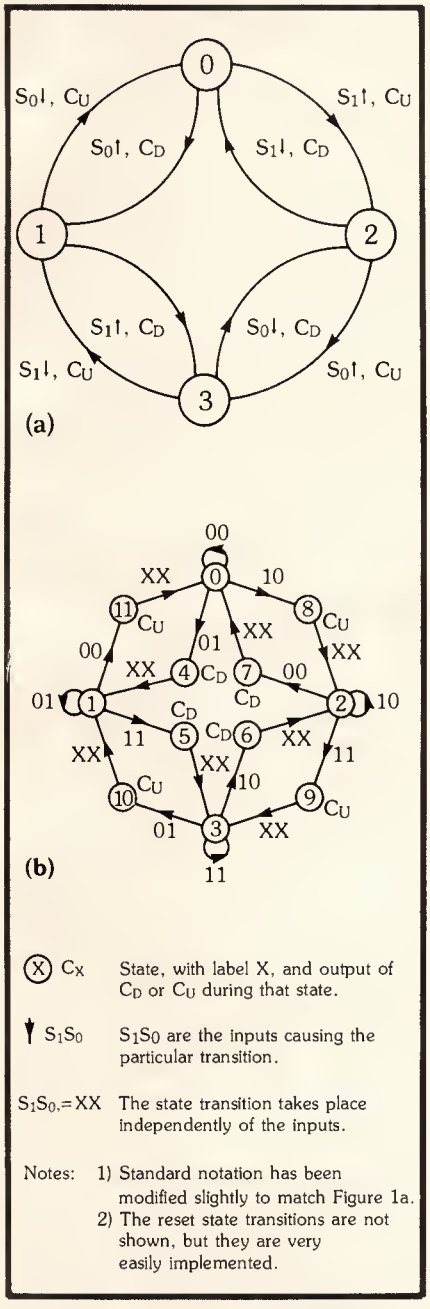


Figure 1. State diagram for the biphase encoder with asynchronous transitions (a); modified state diagram with synchronous transitions (b) (state machine A).

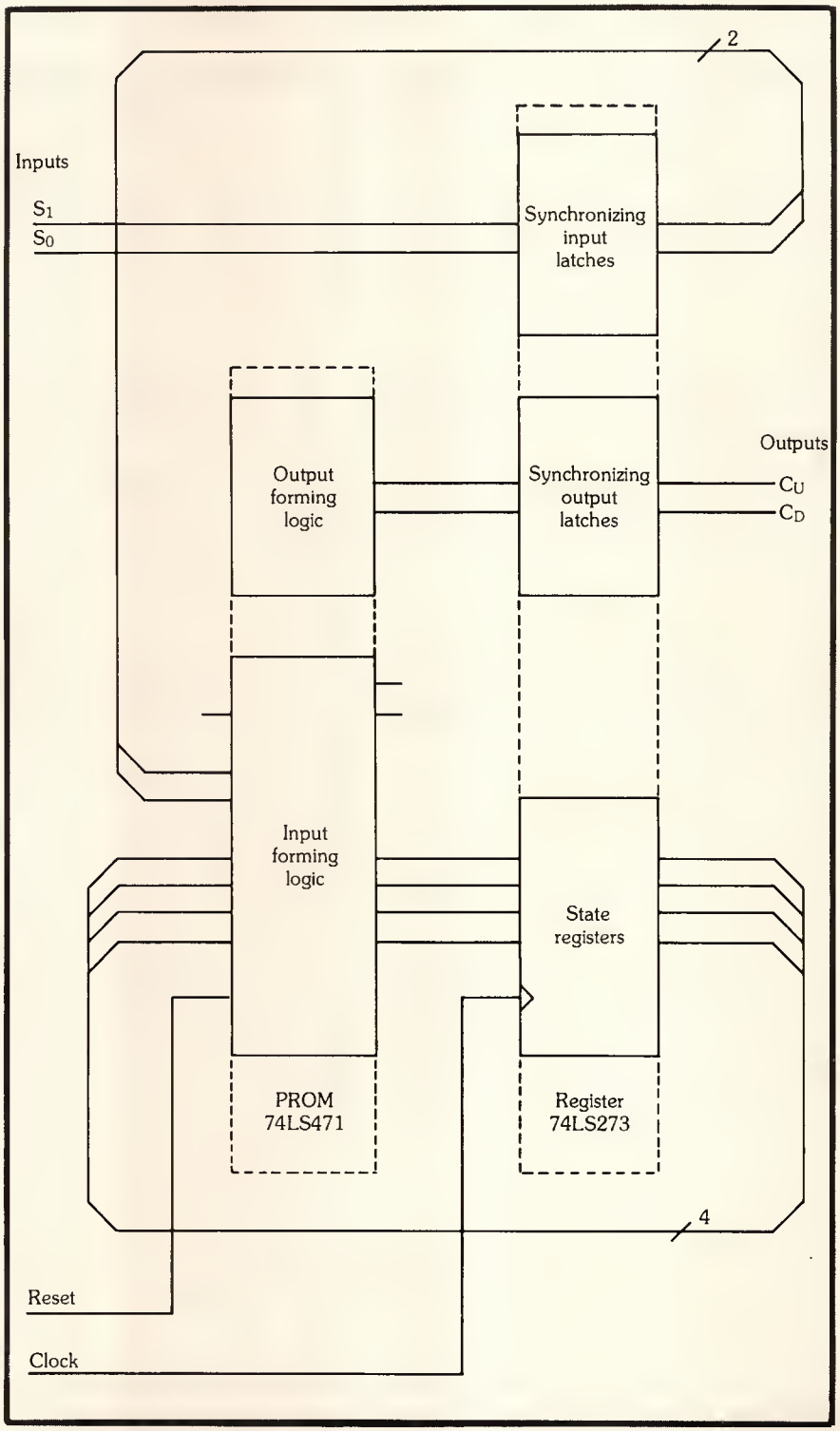


Figure 2. ROM-controlled state machine A (Figure 1b).

machine could be designed. Using standard state machine design techniques, the output-forming logic and latches could be eliminated. Either way, there are spare states that could be used to detect error conditions (i.e., illegal state transitions) and flag a malfunctioning biphase encoder. It might also be important to note that if the biphase encoder outputs were not logically adjacent, then the complexity of the BD controller algorithm would increase, while none of the state machine examples given here would change.

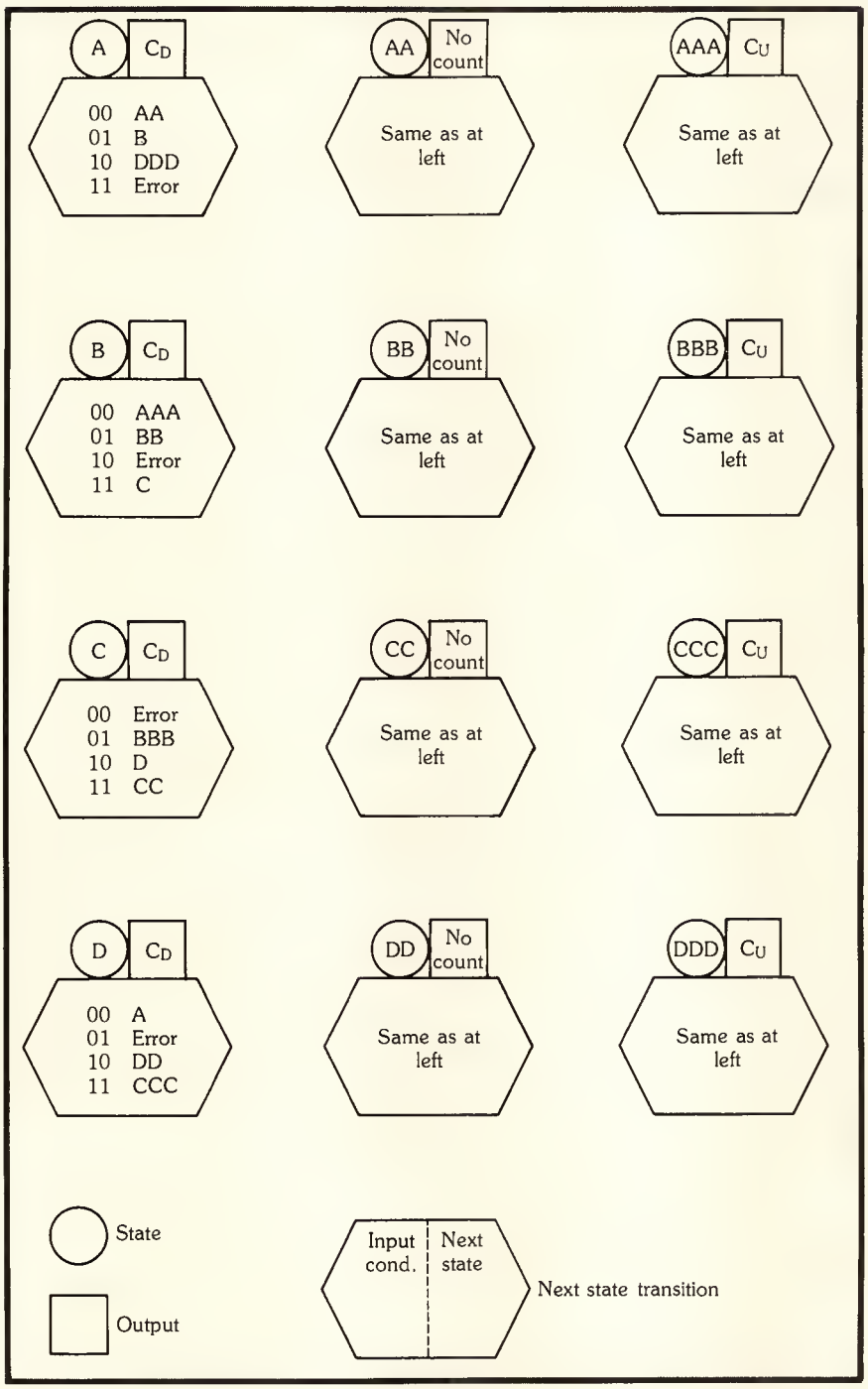


Figure 3. Optimal performance biphase encoder state machine (state machine B).

The full power of state machine design has not yet been exploited. A slightly more complex machine (state machine B)—as shown without transition lines in Figure 3—can be implemented on this same two-chip hardware. State machine B would have all the advantages of state machine A, but with the optimum benchmarks shown in Table 1. Specifically, state machine B is two to three times faster than the BD controller.

The only drawback to state machine B might be that its outputs will not toggle. However, when connecting it to a 74LS169, a binary up/down counter, this is an advantage.

If the designer is willing to spend a few extra hours using boolean algebra, then it is possible to implement state machine B on a single, 20-pin, registered PLA (MMI PAL16R8).

Adding a 15-MHz clock (below spec'ed clock rate) to a single 20-pin PAL implements a controller that is simple, inexpensive, not hardwired, and easy to design and modify. This design will also monitor velocities up to 13,000 meters per second, which is over eight times faster than the limit (1,500 meters per second) of a 10-MHz BD machine. (These comparison velocities assume that one quantum—state change of the encoder output—represents a 1-mm displacement.)

In summary, there are many possible designs between a simple state machine and a 2^n-decision controller. By designing from a generalized state machine perspective, the engineer has a better chance of selecting the optimal solution.

Darryl J. Stewart
724 W. 1720 North, No. 311
Provo, UT 84604

Authors' reply:

Before selecting which bones to pick in Stewart's commentary, let us concede that in our article we concentrated on BDs, microprocessors, and gate logic in roughly that order of emphasis. PLAs (programmable logic arrays) and ROM-based TLUs were mentioned briefly. Multibit finite state automata, or MBFSA, an important class of quasiprogrammable macrologic, were ignored. Admittedly, we have had very little experience with these.

On the other hand . . .

The writer refers to an article entitled "Binary-decision-based Programmable Controllers" and wonders why we did not present "*n*-decision PC" alternatives. The articles were not meant as a precis on various finite-state machine implementations of programmable sequence controllers. We tried to show that one particular machine—namely the BD machine—largely* ignored in commercial applications, can fulfill a useful role. Admittedly the BD is slower, in any suitable application, than dedicated hardwired logic but considerably faster than a microprocessor while retaining similar flexible programmability.

Stewart states that *n*-decision controllers are superior in many applications and would not require any additional logic. This is probably true. Parallel input machines are clearly more efficient than serial ones in both single-bit and string input cases. Two examples illustrate this point:

(1) The biphase encoder supplies two single-bit signals to the BD, S_0 and S_1. Given the previous values of S_0, S_1, it is sometimes necessary to examine only S_1 or S_0 to determine the next state of the biphase encoder. Other times it is necessary to examine both S_0 and S_1. The next state can be any of the following:

C_D, C_U, NC, test S_0, test S_1

By using parallel inputs, S_0, S_1 are always tested together in only one cycle.

(2) Multibit inputs as provided by A/Ds are more efficiently processed in parallel in function computation applications. The correct result is provided in one step rather than *n* steps.

However, the BD machine is the most general type of design for 2^n decision finite-state machines for several reasons:

(1) 2^1-decision machines can be programmed to sample inputs in any order, not just that order defined by input connection hardwiring. This is the beauty of programmable BD logic. The alternative, multibit architecture, makes an implicit, irrevocable commitment to partial bit prioritization and you're stuck with it.

(2) 2^1-decision machines are easier to apply to *m*-input applications than a 2^n-decision machine would be when $n < m$.

(3) Given a 2^n-machine solving an *n*-input boolean function, instructions take the form of ordered minterms of the function. It is not feasible to examine prime implicants, since all *n* bits must be tested even though some are don't-cares. BD machines examine inputs serially, and thus can be programmed only to test prime implicants instead of minterms.

Stewart states that there are many potential modern state machines to choose from and that the designer should not be restricted to using a binary-state machine. This is a "motherhood" statement. The writer mentions several PROM- or ROM-based machines. These are single-use designs that are not easily transported between applications and certainly cannot be multiplexed as the BD is.

The BD is a general-purpose digital PC system composed of software design tools (compiler) and microprocessor supervisor. The BD is absolutely programmable—unlike PROM- or ROM-based controllers which might not be strictly "hardwired" but are definitely "firmwired."

Note that the biphase encoder routine is a typical subset of some larger control algorithm resident in a BD controller. In the current prototype there are 256 16-bit words of program memory, but the encoder routine would occupy only about 26.

Stewart further claims that the BD algorithm would be complicated if biphase encoder outputs were not logically adjacent. If this output was not Gray-coded, two examinations would be required instead of one. This would make the program slower (more instructions to reach a conclusion) but not necessarily longer (more code in all). However, a logically adjacent, cyclic code *must* be used to encode displacement if one is to avoid confusion at quantum boundaries, because it is impossible to obtain perfect registration in free, nondetented mechanical devices. Only a Gray-type code is capable of maintaining a ± 1-bit error everywhere throughout its range of operation. This is why they are exclusively used in multibit shaft encoders and very high-speed A/D converters. See? Even electronic hardware—when pushed to its speed limit—requires Gray-coding to overcome race problems. Though our BD program did not include provision for illegal transitions, this can be done with a few more lines of code. When the BD is used as part of a larger, hybrid control system, illegal transitions would be programmed to trap to a micro interrupt.

The following steps could be taken, however, if one intended to design a machine specifically dedicated to a biphase encoder:

- reduce the instruction set to contain only one kind of input and output instruction, and
- reduce the instruction word to 8 bits.

With the instruction word shown in Figure 1, the entire BD machine (Figure 2) would comprise only two 14-pin packages for input selection and decoding (7 gates); one 14-pin, 5-bit program counter; one 14-pin quad-latch (2 latches required); and one 16-pin DM74S188 32 × 8 bit PROM. The biphase program, including output togglings, is only 26 instructions long. It would fit quite nicely within the range of the 5-bit in-

*This would have read "completely" but for the strong objections of Mange et al. (see earlier letter).

struction field provided for addresses. But this is not our primary goal. A hardwired VLSI implementation would be faster, occupy less real estate, and would perhaps be cheaper in quantity. This decoder was only presented as an example of a single, easily understood BD application. As stated previously, this BD decoder algorithm is a typical routine which might commonly appear as part of, say, a larger control program. There is enough program memory to implement a reasonably elaborate procedure without resorting to any extraneous hardware.

When we started, the issues seemed to be quite clear:

(1) how to overcome the inconvenience of using 8-bit microprocessors to implement sequence controllers whose inputs and outputs cannot be naturally associated to form numbers,

(2) how to preserve the flexibility of microprocessor-like instruction sequences, and

(3) how to compensate, at least partially, for the inherent slowness of examining binary inputs one by one and, similarly, of generating the binary outputs.

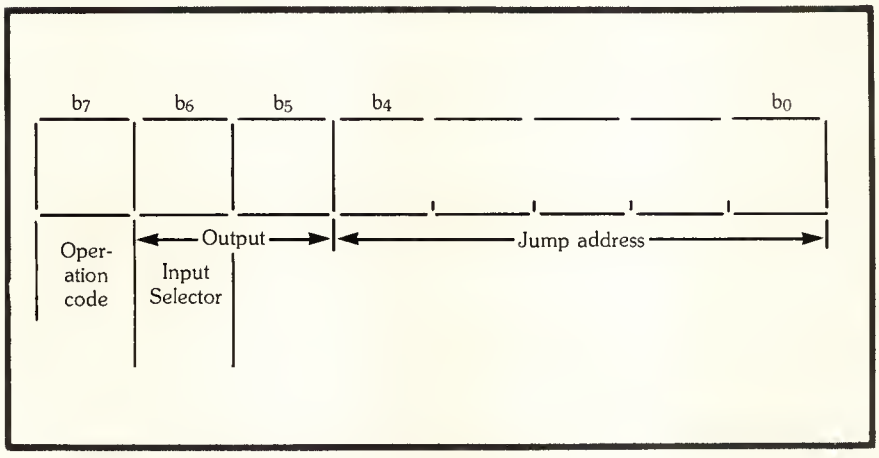


Figure 1. Instruction word reduced to 8 bits.

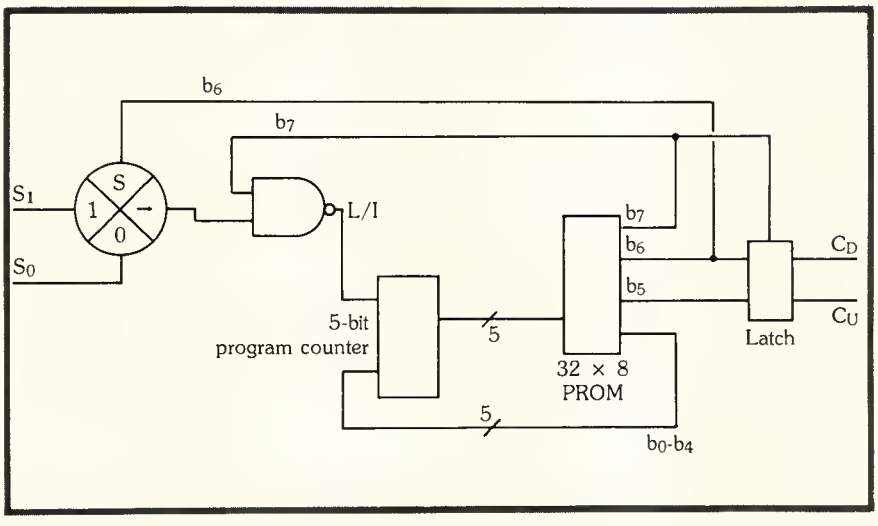


Figure 2. BD machine resulting from use of the reduced instruction word.

We still maintain that the programmable BD machine with parallel outputs is a good compromise. It can execute any programmable procedure. It can be made to look like any *m*-input, *n*-output device. For simple procedures, it's much faster than a conventional microprocessor. At no time did we claim that it was faster than a well-tailored, multibit finite state machine. We do believe, however, that it is much easier to program. Apparently our implicit assumption that the designer had a choice only between boolean and BD design techniques was overly simplistic. Maybe the situation is better illustrated by a spectrum with microprocessors at one end and hardwired logic at the other, while BD and MBFSA are somewhere in the middle.

This is getting to be a bit like the five blind, wise men of Hindustan who went to see the elephant. Each gropingly identified a specific attribute of the beast and loudly proclaimed his limited viewpoint to be complete and uniquely correct!

It's somewhat unfortunate that we chose to demonstrate BD effectiveness with a single example—the biphase encoder—but that's the only application device which was working when we wrote the article. We led with our collective chins. Although it isn't fair to mention unpublished and largely untried BD application, a short "notebook" of generally useful BD programs and their trees has been prepared. Included are things such as adders, comparators, first- and second-differencers and short, straight and curved trajectory generators. We feel that such macros are the beginnings of a high-level BD language. Prospects for similar evolution of MBFSA software may exist, but we haven't studied them nor do we know anyone who has.

P. J. Zsombor-Murray
L. J. Vroomen
R. Hudson
DATAC Computer Lab
McGill University
Montreal, Que. H3A 2K6

Reader Interest Survey

Indicate your interest in this department by circling the appropriate number on the Reader Interest Card.

High 173 Medium 174 Low 175

MicroLaw

by Richard H. Stern/Law Offices of Richard H. Stern/2101 L Street NW, Suite 800/Washington, DC 20037

Proprietary rights in cell libraries

The economic importance of cell libraries and gate arrays is increasing. It is estimated that, within six years, cell library ICs will become the major application-specific integrated circuits (ASICs) in sales volume, with projected annual sales of $2.8 billion.[1] The growing importance of chips produced by cell library and gate array systems has stimulated interest in who owns the proprietary rights to the systems and their ASIC products.

Software. The software used to lay out cells and route their interconnections, and the related documentation, is ordinarily proprietary to and copyrighted by the cell library licensor. This protection is often supplemented by confidentiality or trade secret agreements. The routing algorithms themselves and the ideas embodied in the software are not protectable under copyright law. They may be protectable as trade secrets, however.

Generally speaking, the same considerations apply to gate arrays. They also apply to mask-programmable ROMs and programmable logic arrays.

Cell databases. The information about the individual cells of cell libraries is protectable in part by copyright, and to the extent that it relates to cell topography (layout within the cell) is protected under the new Semiconductor Chip Protection Act of 1984 (SCPA).[2] There is a serious problem in securing cell topography protection at this time, however, because the US Copyright Office has issued regulations, in effect, repealing the new law insofar as it concerns cells. That point is discussed below.

Chips. One of the most challenging questions is who owns what rights to the ASICs produced by use of cell library or gate array systems. Customers generally concede that they do not acquire any rights to cell libraries and cells, to unpersonalized gate arrays, or to the software used to produce customers' cell and gate array layouts. The controversy is over who owns the rights, if any, to the chip "ensemble," or the topography (layout) of the ASIC considered as a totality. Customers attempt to assert as much exclusive right as they can to what they consider "their" ensemble, while licensors try to avoid making concessions that will hamper their subsequent business activity.[3] These interactions are discussed below.

How the SCPA Applies to ASICs, Cells, Gate Arrays

The SCPA provides a 10-year term of protection for "semiconductor chip product" layouts that are original and are not commonplace or staple. During the 10-year period, no unauthorized person may make, sell, or import the IC, and no one may induce or "knowingly cause" the manufacture, sale, or importation of the IC. In this connection, making masks for a pirate would be a violation of SCPA rights, and preparing database (pattern generation) tapes would be in the same category. To secure protection under the SCPA, the owner of rights to the layout of a chip, cell library, cell, or gate array must file application for registration with the Copyright Office within two years of first commercial exploitation (such as beta testing); in the case of products first commercially exploited before November 8, 1984, however, the deadline is June 30, 1985.

Intermediate forms. The term "semiconductor chip product" means the "final" or "intermediate" form of an IC.[4] An intermediate form is any transitional form of the die, that is, one short of final form. The intermediate forms of a semiconductor chip product are typically seen on partially fabricated wafers. An example is a wafer where there is a temporary oxide layer on the surface, with openings to permit dopants to be beamed into selected parts of the silicon, after which the oxide will be removed. Another example, of more interest in the context of the present discussion, is an "unpersonalized" gate array or mask-programmable ROM (one with no upper metal layers interconnecting the elements), which is a form of the product in which wafers are stockpiled until a customer orders their "personalization." The unpersonalized layers of the final semiconductor chip product are the ones into which the greatest design ingenuity and investment has gone, and gate array firms are quite interested in preventing competitors from copying this form of the product.

Registration of gate arrays. Since the SCPA permits the registration and protection of a "semiconductor chip product," and that term includes unpersonalized gate arrays, they may be registered. A problem has arisen, however, with the Copyright Office. It has issued a regulation that "registration will be refused for the mask work [chip topography] as fixed in an intermediate form if fixation in a final form has occurred before registration is sought."[5] That means that a gate array firm cannot protect the unpersonalized form of a gate array if the firm has ever made the gate array in a "final" form, that is, with a metal, personalization layer. Since it is not ordinarily possible to tell whether a gate array design works until it has been fabricated with metal interconnections and tested, all gate array makers prepare a gate array with metalization and test it before finalizing the design and making thousands of wafers. As far as the Copyright Office is concerned,

however, doing that is inimical to the policies of the SCPA; and thus it bars registration. The semiconductor industry has protested to the Copyright Office and urged that it reconsider.

Cell libraries. The legislative history of the SCPA several times has indicated Congress' desire to protect cells and cell libraries. Thus, the House Report states that a cell is protectable if it meets the originality requirements of the new law.[6] The Senate Report states that copying an individual cell forming part of a cell library is an infringement when the cell layout is not dictated by the function of the cell and the copying is close enough for the later cell to be "substantially similar" to the first.[7] Clearly, Congress was informed about the economic importance of cell libraries and expected them to be protected under the new law.

Fixation in final form rule. Yet, as in the case of gate arrays, the Copyright Office has issued regulations the effect of which is to deny registration of cells in most instances. First, the Copyright Office applies to cells the same rule discussed earlier as to gate arrays—that the cell library proprietor must not make in final form any actual chip, that includes the cell prior to registration, upon pain of losing the right to register the cell. Apparently, the Copyright Office wants to register cells only before the designer has had an opportunity to ascertain whether the cell works.

The 20 percent rule. Second, the Copyright Office has issued a further regulation that applies particularly to cells. No "intermediate form" of a semiconductor chip product (which the Copyright Office for unexplained reasons considers to include cells) may be registered if it amounts to less than 20 percent of a final semiconductor chip product.[8] The Copyright Office does not indicate in what terms the measurement is to be made—die area, volume, number of transistors, power dissipation—but actually that makes little difference. Many or most cells amount to less than 20 percent by any measure. Indeed, at least one of the cells mentioned in the legislative history of the SCPA, an oscillator,[9] would not meet the Copyright Office's 20 percent rule.

The Copyright Office does not explain why it fastened on the number 20, but it does state that it generally disapproves of the registration of cells because it permits cell library proprietors to register more than one aspect of a particular chip. The Copyright Office feels that doing so tends to tip the balance in favor of chip proprietors in litigation over infringement of mask work rights, and decreases the effectiveness of the SCPA's exemption of reverse engineering from liability under the new law.[10] Whether that is so is questionable. But irrespective of whether it is, under our system of government Congress, not the Copyright Office, writes the law, and the courts, not the Copyright Office, interpret it. Again, the semiconductor industry has protested to the Copyright Office against the regulation, but the Copyright Office seems to feel that anyone who disagrees with the 20 percent regulation simply can sue the Copyright Office: "The [Copyright Office's] refusal to register may of course be tested in the courts against... the Office itself."[11]

Problems with registering entire ASICs

Part of the Copyright Office's thinking seems to be that cell library owners, instead of registering their cells, should register entire chips in which are configured the cells to be protected. But that is not a satisfactory alternative for several reasons.

Proving infringement of mask work rights. First, it is more desirable to have a registration on just a valuable cell itself than it is on a whole chip with irrelevant and possibly unprotectable old material included in it. Then, if there is a dispute over copying, the chip proprietor can point to a situation of 100 percent copying, instead of saying that the defendant took the little gizmo in the lower left hand corner of the registered chip layout.

The "thicket." Second, some cell library proprietors feel that it is competitively more valuable to have as many registrations as possible, because a rival firm or piratical chip customer is then more likely to hesitate to copy. When the competitor has to face only one patent or copyright or mask work right, or a few, the competitor may feel that a risk-benefit analysis justifies running the risk of infringement liability. But when there is a whole intellectual property "thicket" to face, the mere cost of analyzing the risks may be so high that it is not worthwhile to do so; moreover, the likelihood of a defendant's success in infringement litigation is bound to be lower when he faces 30 pieces of intellectual property rights than when he faces only three.[12] Some observers, however, feel that the "thicket" strategy puts competitors at an unfair disadvantage and therefore disapprove of it.

Section 902 (b) and Catch-22. Third, there is a "Catch-22" problem. The Copyright Office says, Go register the whole chip. But the statute may say, You cannot register a chip developed from a cell library. The problem may be even more acute with gate arrays. SCPA § 902 (b) requires that a chip layout must be "original" and also must not be "staple, commonplace, or familiar in the semiconductor industry" when the combined layout is "considered as a whole." What this controversial language means is very unclear. It seems to call for some degree of novelty—not as much as the patent system requires, but still something beyond mere failure to replicate a preexisting layout.

The design of an ASIC from a cell library is a computer-effectuated layout procedure. The software associated with the cell library combines preexisting cells in the cell library, in conjunction with the functional specifications of an end-user, to produce a layout output. Is that an action that is novel, original, or creative enough to satisfy the requirements of SCPA § 902 (b)? Nobody knows with confidence. To be sure, this is a fact-dependent question. If the software is imperfect enough to require a great deal of human "massaging" and "tweaking" of the output from the computer, these acts may overcome the originality/ commonplace problem. But if the cell library system is a very good silicon compiler, there may be insufficient human (non-computer) effort to satisfy SCPA § 902 (b). Perhaps our cell library technology is still imperfect enough to permit escape from the Copyright Office's Catch-22.

The problem is worse for gate arrays, however, because gate array software is relatively better than cell library software, in the sense that the output from gate array software may need virtually no massaging or tweaking to work satisfactorily as a layout for an ASIC. Hence, if the unpersonalized part of a gate array cannot be registered and thus

protected, it may be impossible to protect the gate array at all. Catch-22 would overcome the promise of Congress to protect chip creativity. There would be a triumph of bureaucratic logic over common sense. (In the case of mask-programmable ROMs, the problem is still worse.)

There are arguments that can be made on either side of this issue, and the controversy goes beyond registrability. It also impacts the relationship between cell library licensor and customer, since it affects who owns the rights to the ASIC as a whole. Rather than address the issue only in the context of the semiconductor industry-Copyright Office dispute over registrability of cell libraries and unfinished gate arrays, we may more profitably turn to the entire controversy over who owns what rights in the final ASIC, considered as a whole. But it should be noted, now, that if the Copyright Office's regulations stand unchallenged, there may be no way to protect most ASICs. The cells of cell library ASICs and nonfinal gate arrays will be unregistrable, while the final ASIC products may be held to lack the "nonstapleness" requisite to protection under the SCPA. That is a worst-case scenario, of course. The Copyright Office may back down on its anti-ASIC regulations, or someone may get a court order compelling it to do so. Alternatively, the courts may be persuaded that the human or nonhuman creativity involved in using a cell library system is adequate to meet the requirements of SCPA § 902 (b)'s "nonstapleness" test. The last possibility falls within the discussion that follows.

In the meantime, the owner of a cell library or gate array system should protect his position by filing applications for registration before rights become permanently forfeited. The deadline for filing is two years after first commercial exploitation, in general, but before July 1, 1985 for layouts first commercially exploited before November 8, 1984. Failure to file will result in voluntary forfeiture of all rights. Filing an application increases the system owner's options, because filing opens up the alternative possibilities that (1) the Copyright Office may relent on its position and (2) otherwise, the owner can sue the Copyright Office to compel it to register the owner's cell, unpersonalized gate arrays, or similar products. (The same considerations that apply to mask-programmable ROMs and programmable logic arrays also apply to gate arrays.) Given the fact that the Copyright Office's anti-ASIC position is flatly contrary to the legislative history of the SCPA and common sense, it would seem reasonable that the Copyright Office will eventually voluntarily or involuntarily back down.

What is a cell library?

A cell library is a system used to design and lay ut application-specific integrated circuits (ASICs) in accordance with the functional requirements of particular end users. A cell library has two principal components:

- A database containing information about cells, which are modules used in assembling ASICs. The information includes the size and topography of the cells, heat dissipation, necessary connections, and electrical specifications.
- Software for placing cells on the die of ASICs and for routing their interconnections economically and in accordance with design rules.[1] There is additional software for simulating performance of the resulting ASICs.

How does this work? By way of illustration, one could imagine the case of an architect who has a proprietary system with stored plans for kitchens, living rooms, bedrooms, bathrooms, and so on. When a customer comes to the architect and states, "I want a house with one master bedroom, two bedrooms for my two children, a large kitchen, a living room, and a recreation room," the architect inputs the customer's functional requirements into the proprietary system. It outputs a set of plans in which the required rooms, with necessary hallways, are layed out; the plumbing, heating, and electrical ducts and connections are shown; and so on. When a second customer comes in with a different sized family and different functional requirements, a different set of plans is produced. Of course, if a third customer also has two children and likes large kitchens, the plans that the system produces may be similar to those for the first customer.

A cell library transaction may fall into any part of a spectrum of allocation of engineering efforts between cell library proprietor (or "licensor") and customer (or "licensee"). At one end of the spectrum the customer supplies only a list of functional specifications (and perhaps a logic diagram) for the chip, and the cell library proprietor does all of the design work with the cell library system. At the other end, the customer uses the cell library and software to design a prototype chip.[2] The price that the customer pays depends on the relative allocation of work between the customer and cell library proprietor. Ordinarily, the latter actually manufactures the ASIC for the customer, acting as a silicon foundry.

References

1. That is a shorthand way of saying: manipulating the database in accordance with computer programs to produce an output set of data, which can be used to make a pattern generation tape for masks for manufacture of ASICs in accordance with the semiconductor process technology of the cell library proprietor.
2. An important part of the process, omitted from the preceding description, is computer simulation of the operation of the chip that resulted from the use of the cell library, prior to fabricating it. This may involve checks for timing discrepancies, excessive heat output, and other potential flaws in the design.

Licensor-Customer Conflicts Over Rights

That is the legal background in which cell library licensors and their licensees or customers operate. The legal uncertainties discussed above interact with ordinary business considerations to raise interesting problems over who legally can own what, who owns what in the absence of an agreement providing otherwise, and what kind of cell library agreements should the parties enter into.

The cell library licensor wants to be free to license the use of the cell library without worrying about disputes when a later customer's use of the system produces a chip that is similar to the chip that an earlier customer's use of the

system produced. But the earlier customer is concerned about pirates and wants as much protection as possible against either piracy from an unrelated third party or competition resulting from a subsequent licensee's use of the cell library system.

The earlier customer therefore demands as much of a "property right" in the ASIC as possible. The only useful property rights available, for all practical purposes, are those under the SCPA. One possible right that might further the customer's position is the right in the ASIC considered as a whole, apart from any rights in the cells configured within the chip. In negotiating an ASIC deal, the customer may therefore ask the licensor to assign ownership of all mask work rights under the SCPA, in the ASIC, to the customer.

Assignments of ASICs. One question is whether there is any "ensemble" right, or right in the ASIC as a whole, given the "nonstapleness" problem discussed above. If there is none, the cell library licensor would be assigning a nonexistent right to the customer. Putting that question aside, a second question is whether the cell library licensor will divest himself of his own rights in the individual cells by acquiescing in the customer's request that the licensor assign (or try to assign) "ensemble" or other rights in the ASIC to the customer. That issue presents, in the first instance, a difficult drafting problem. The problem is like that of a homeowner who wants to sell the house considered as a whole but not the individual rooms in the house. It is difficult, but not impossible, in terms of drafting techniques.

But a further problem is whether it is legally possible to have separate ownership of SCPA rights in the cells and in the ASIC as a whole. The Copyright Office has taken a rather strong position on the indivisibility of ownership of rights under the SCPA. Several members of the semiconductor industry have protested against the regulations, and the Copyright Office may become convinced that it has misinterpreted the SCPA. Probably, such rights are more divisible than the Copyright Office thinks, but there is some uncertainty as to what the consequences will be if one attempts to fragment ownership rights in a semiconductor chip product layout. It is possible that no one will be the owner any more, and that both parties to the transaction will become exclusive licensees; that may well not be so, but it is an unsettling possibility, and it would create problems.

Uncertainty of this kind makes many cell library proprietors unwilling to assign or purport to assign the "ensemble" or other rights in ASICs to their customers, and makes them prefer to grant exclusive or nonexclusive licenses. From the perspective of negotiating position, a cell library or gate array system owner should strongly resist any assignment of its rights and agree only to a license, not only for this reason but for a number of others concerning relations with subsequent licensees and third parties.

Licenses of ASICs. Under a *nonexclusive* license, the customer simply has the right to use and distribute the chip (ordinarily as part of equipment) and also, assuming that the license would be understood to give a right to have the chip made, the right to have the chip (along with its constituent proprietary cells) second-sourced. Price or royalty terms may impose minimum purchase requirements before second-sourcing. There may be technical assistance (such as use of tapes) associated with authorized second-sourcing. The customer acquires no right to force the owner of the cell library (or gate array) system into litigation with third parties.

Under an *exclusive* license, a customer *A* has, at minimum, the licensor's assurance that he will not use the results of the cell library license with *A* (such as the photomasks and pattern generation tapes for the ASIC) to manufacture the same chips for *B*. Customer *A* can probably sue alleged infringers without consulting the cell library (or gate array) system owner, and may otherwise be able to force the owner into infringement litigation. Absent specific agreement on details, it may be uncertain what control, if any, the licensor is expected to exert over other customers who use the same cell library system. In the case of systems houses that use the cell library system under license to create ASICs for their own customers, it may be very difficult to ascertain (let alone police) what goes on.

A further problem with exclusive licenses is that it is very uncertain under the SCPA whether independent use of a cell library system by a later user *B*, whose functional specifications for an ASIC are deliberately or inadvertently similar to those of an earlier user *A*, makes *B* liable to *A* for infringement if the later ASIC of *B* is very similar to the earlier ASIC of *A*. When there is a nonexclusive license to *A* the question does not arise, since a nonexclusive licensee has no exclusive right to the layout; but when there is an exclusive license to *A*, the liability of *B* and also that of the cell library licensor to *A* is uncertain. If more or less "independent creation" of the chip is a defense, there may be no liability to *A*. If "independent creation" is no excuse, or if inputting specifications obtained by reverse engineering the earlier chip negates "independent creation," then there may be liability. An outline of the arguments on either side of the controversy and comments on them are set out below.

Avoiding disputes by agreements about ASIC rights. Since the law is unclear, it is important not to leave the matter up to the happenstance of how a court will interpret the SCPA. The alternative to rolling dice is to enter into an agreement that spells out and governs the parties' mutual rights and obligations. But that is more easily said than done, and trying to do so can create more present business problems than the risk of future problems may make it appear worth. That is, a dispute over how to settle the matter could break a deal. A suboptimal resolution of the question (e.g., ignoring it) may never be tested by harsh events. There is also a large zone of reasonable terms of agreement within which the parties can allocate rights, if zeal in bargaining does not overcome common sense. For example, an exclusive licensee may secure a promise from the cell library licensor that SCPA rights will be enforced against any third party pirate or anyone else except a *bona fide* licensee who actually uses the cell library system to create a similar ASIC, with or without previous reverse engineering efforts. Probably, both somewhat more and somewhat less of a promise than that would be in the zone of reasonable terms.

What happens absent any agreement? Suppose the problem is wholly brushed under the rug, however, and the agreement of the parties says nothing about SCPA rights or the ASIC "ensemble." Suppose the problem of *A* and *B*, described above, occurs. Who, if anyone, is liable to whom for what? The customer *A* would argue that he created the ASIC and is entitled to the ensemble right, that *B* committed infringement of

A's rights under the SCPA, that the cell library licensor was also responsible for the infringement, and that they both owe damages to *A* and/or should deliver up to *A* all the profit they made on the ASIC.

In that case, the second user *B* would claim that there is no "ensemble" right here, because an ASIC whose layout a computer made "mindlessly" from preexisting cell data and preexisting software is not original and/or is staple or commonplace. *B* would also argue that his independent creation and/or reverse engineering excuse the substantial similarity between the ASIC layouts. The cell library licensor would argue that either there is no "ensemble" right or else it belongs to the licensor because his cell library and software did all the creative work in laying out the ASIC.[13] The licensor would also deny that merely licensing the cell library to a licensee is an infringement or contributory infringement of any rights under the SCPA (this would be correct). The licensor might also argue that his making *B*'s ASIC as a silicon foundry was not infringement, or was innocent and excusable, because the licensor did not know that customer *B* was infringing customer *A*'s alleged rights; this argument would probably be rejected as incorrect.[14]

What is a gate array?

A gate array is another ASIC, which consists of an array of gates on a chip. The first approximately seven or eight layers of the chip contain unconnected, isolated gates. The top one or two "personalization" layers of the chip interconnect the gates in accordance with an application-specific pattern.

The gates may be simple NANDs or NORs. They may instead be complex gate macros of as many as several dozen transistors, which the user may interconnect in a variety of ways, to make multi-input ANDs, flip-flops, adders, and other logical units.

As in the case of cells, the gates on a chip (and gate macros) are interconnected in a pattern determined by software proprietary to the gate array seller. Because there are fewer possibilities for gates than for cells, the probability is greater that the gate array system will produce a similar pattern for different customers who have similar functional specifications.

The preceding discussion shows that there is considerable uncertainty under present law as to who owns what in ASICs and who can be liable to whom for making and selling ASICs. It also indicates that there are a number of steps that can be taken to reduce uncertainty, to maximize negotiating position if a disagreement arises, and to increase one's likelihood of success in the event of litigation.

First, a cell library system owner should promptly file applications to register his cells, and a gate array system owner should similarly register his unpersonalized gate arrays. The Copyright Office may resist, but the cost-and-benefit tradeoffs indicate that this is more prudent than simply acquiescing voluntarily to permanent forfeiture of these rights. Owning a registration of SCPA rights in one's cell library creates a number of negotiating advantages and is an absolute condition precedent to suing anyone for copying.

Second, in negotiating an ASIC deal, the cell library owner should refuse to assign anything, should be reluctant to give an exclusive license, and should propose a nonexclusive license; the customer may open by proposing (but should not seriously expect) an assignment of all mask-work rights in the ASIC or at least its "ensemble," should try to get an exclusive license on the ASIC, and should resist but be prepared reluctantly to acquiesce in a nonexclusive license. In a nonexclusive license, the customer should try to get an assurance that the cell library owner will sue pirates and should be prepared, in return, to negotiate who pays for any lawsuit. The cell library owner should be wary of any provision that may cause it to be dragged into litigation against alleged pirates, particularly when they may be other licensees of the cell library owner.

Finally, it may not be feasible or prudent to try to get all of these things spelled out in the parties' contract, because it may unduly delay or hinder consummation of the deal. But it is prudent for each party privately to review these considerations, so that it can make a more realistic appraisal of the value and risks of the deal—which may impact the party's unstated bottom line for negotiation on price.

References

1. *EDN,* "Guide to Semicustom-IC Technology and Application," Vol. 29, No. 4, 1984, p. 93. In 1982 cell library-ASIC sales were an estimated $30 million. Sales projections for gate arrays in 1991 are $2 billion, and 1982 sales were estimated at $130 million. Among the business advantages of using cell library or gate array ASICs instead of full-custom ASICs are much lower design cost, shorter turnaround, and possibly greater reliability. In the case of at least gate arrays, and possibly also cell library chips, production cost is lower and yield higher because the product is farther along the learning curve than a full-custom chip can be. On the other hand, a full custom chip will generally have a smaller die, will execute faster, and in large enough volume becomes cheaper than a semicustom chip.
2. For a brief description of the new law, see *IEEE Micro,* Feb. 1985, pp. 74-75. For a more detailed, book-length discussion, see R. H. Stern, *Semiconductor Chip Protection,* Harcourt Brace Jovanovich/Law & Business, New York, 1985.
3. Customers are also zealous in asserting rights to their functional specifications, logic diagrams, schematics (if any), and lists of parts or elements to be included within the chip. (The term "net list" is sometimes applied to the list of components within an ASIC. But the term is ambiguous.) It is unclear whether, and to what extent, these things are protectable against third parties. To the extent that they can be learned by reverse engineering, they are probably unprotectable against competitors. However, they may be protected against a person who obtains the information by improper means (espionage) or uses it in violation of an agreement to treat it as confidential.
4. See SCPA § 901 (a) (1).
5. 37 C. F. R. § 211.4(e) (2), 50 Fed. Reg. 272 (Jan. 3, 1985).
6. House Report No. 98-781 (1984), pp. 26-27.

IMSL Quality

Mathematical and Statistical FORTRAN Subroutines

for IBM Personal Computers

Today's personal computers have the power and sophistication for work in science, engineering, and other technical fields. Now IMSL has developed MATH/PC-LIBRARY, STAT/PC-LIBRARY and SFUN/LIBRARY — software equal to the task of serious mathematical and statistical FORTRAN programming on IBM personal computers.

MATH/PC-LIBRARY
Subroutines for a wide variety of mathematical applications.

STAT/PC-LIBRARY
An efficacious selection of statistical subroutines.

SFUN/LIBRARY
The most comprehensive subprogram library available for evaluating mathematical and statistical special functions.

A Step Beyond other FORTRAN Libraries

Unlike other PC-compatible FORTRAN libraries, these versatile resources are part of IMSL's integrated system of FORTRAN libraries, offering a uniform approach to problem solving across a wide range of computer types and sizes. The IMSL libraries are ideally suited to today's multiple-computer environments, providing accurate results whether you're using your PC or a supercomputer.

With MATH/PC-LIBRARY, STAT/PC-LIBRARY and SFUN/LIBRARY you can select complete, thoroughly tested subroutines instead of writing them. You'll reduce development time, decrease maintenance costs, and enjoy the accuracy and dependability that have made IMSL a world leader in advanced numerical software systems.

World-Class Software Resources for Your PC

MATH/PC-LIBRARY and STAT/PC-LIBRARY contain the most frequently-used subroutines from the IMSL Library — a resource chosen by corporations, universities, research centers and governments in more than 50 countries. SFUN/LIBRARY offers a versatile set of subprograms for evaluating mathematical and statistical special functions.

Formerly available only for mainframes and minicomputers, these subroutines can now expand the programming capabilities of your personal computer.

MATH/PC-LIBRARY and STAT/PC-LIBRARY are available for IBM PC, PC XT, PC AT, and Portable PC systems running either IBM Professional FORTRAN or Microsoft FORTRAN 3.20. SFUN/LIBRARY is available only for IBM Professional FORTRAN environments.

IMSL
Problem-Solving Software Systems

Reader Service Number 1

For complete technical information, return this coupon to IMSL, NBC Building, 7500 Bellaire Boulevard, Houston, Texas 77036 USA. Telephone: (713) 772-1927. Telex: 791923 IMSL INC HOU. In the U.S. (outside Texas), call toll-free 1-800-222-IMSL.

Send complete technical information about:
☐ MATH/PC-LIBRARY ☐ SFUN/LIBRARY
☐ STAT/PC-LIBRARY

Name

Dept. Title

Organization

Address

City/State Code

Area Code/Phone

Telex

Computer Operating System

FORTRAN Compiler/Version IEEEM65

Copyright © 1985 IMSL, Inc.

7. Sen. Report No. 98-425 (1984), p. 17.
8. 37 C. F. R. § 211.4 (e) (2), 50 Fed. Reg. 272 (Jan. 3, 1985). Although this rule is primarily directed against cells of cell libraries, it also prevents registration of gate-array gates and gate macros.
9. House Rep., p. 26, n. 51.
10. SCPA § 906 (a) provides that it is not infringement of mask work rights to make a competitive chip by reverse engineering, unless the resulting chip is "substantially identical" to the original chip.
11. 50 Fed. Reg. 266 (Jan. 3, 1985).
12. Other factors in some cell library proprietors' thinking are that building a thicket of cell registrations is too costly to make it an effective strategy, and that too few individual cells are technologically advanced enough to justify making an effort to prevent competitors from copying them.

 That is probably an incorrect cost-benefit analysis, however, based on the high unit cost of the first several cell or chip registrations. It costs possibly $500 or more in total personnel time to do one's first chip registration, but the learning curve is very steep. At 100 units and beyond, the incremental cost of volume production of applications for registration of cells or even whole chips probably drops assympotically toward approximately $50 per unit.

 On the second point, to focus on the technological merits of individual cells is to disregard the cumulative business effect on actual and would-be competitors of the registration of the cell library *en masse*—i.e., the "thicket" effect—which is to entrench the position of an incumbent against any such competitor.
13. To that, *A* would reply that his work in creating the functional specifications counts more heavily. The cell library licensor would counter that functional specifications are not protected by the SCPA, and only layouts are; moreover, these specifications were ascertained by reverse engineering, which SCPA § 906 (a) specifically authorizes. Doubtless, *A* would have a rebuttal to all of that.
14. The SCPA has an innocent infringement defense, SCPA § 907, but it applies only to chip purchasers and resellers, such as chip brokers and manufacturers and sellers of IC-containing equipment. The innocent infringement defense does not apply to chip manufacturers, such as silicon foundries. (There is no innocent infringement defense in patent law, either, and book printers have long been held liable for printing infringing books regardless of their ignorance of the author's copyright infringement.

Reader Interest Survey

Indicate your interest in this department by circling the appropriate number on the Reader Interest Card.
High 170 Medium 171 Low 172

MicroReview

Editor: David L. Hannum/AT&T Information Systems

Short Reviews

by David L. Hannum

Hardware

Juki Model 6300 Daisywheel Printer

The Juki 6300 is a low-cost, letter-quality printer that has excellent print characteristics and a useful user manual too boot. When I tested the Juki's print quality against those of the more expensive Qume and Diablo printers, I was unable to tell the difference. Although I could not gauge the overall durability of this machine, I did notice the noise difference between my reasonably quite dot matrix printer and the rather noisy daisy wheel printer. This noise, however, is not uncommon among printers of this type.

The user manual is quite helpful, particularly to owners of IBM products; and, from what I am told, future editions will keep up with the trade. Additionally, the manual describes the operational setup for some of the more popular software programs, such as Lotus 1-2-3, Multimate, and Wordstar. Even the procedures to set up the printer with your DEC Rainbow or Apple computer and compatible software are given ample space. The one omission I noticed was Microsoft's Word, which I happen to use.

This is one piece of equipment that gets a "very good" for ease of learning, ease of use, and overall quality.

Books

Buyer's Guide to Modems & Communications Software by Terry C. Silveria, Sanjiva K. Nath, and Edward Hogg (Tab Books, 1985).

This guide begins well enough with a simple overview of data communications, couched in layman's terms and intended

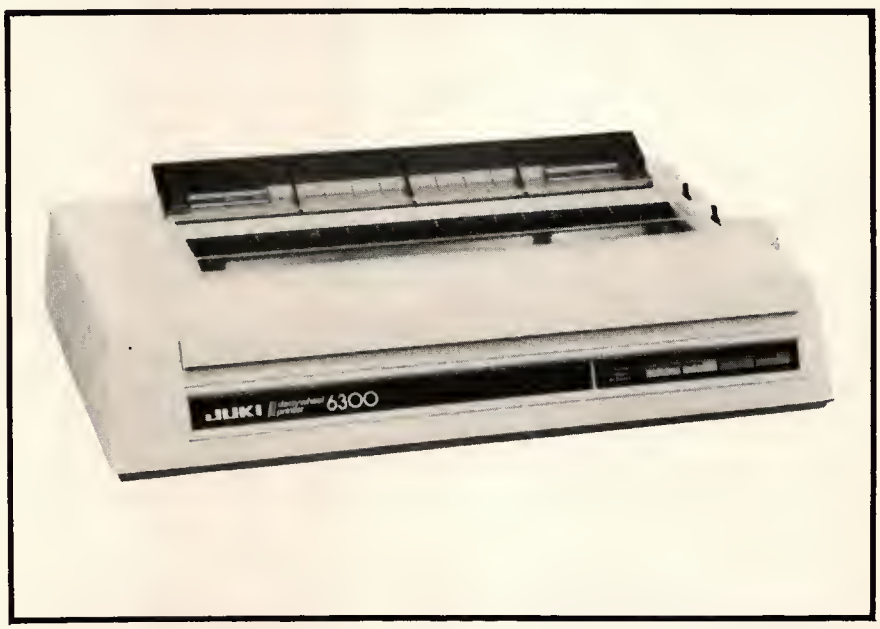
continued on p. 82

Printer at a Glance

Juki Model 6300 Daisywheel Printer

Juki Office Machine Corp.
23844 Hawthorne Blvd.
Suite 101
Torrance, California 90503
(213) 320-4860

Print speed	40 cps
Print wheel	Diablo 96 character plastic
Number of characters per line	132 (10 pitch) 158 (12 pitch) 112 to 393 (proportional spacing)
Maximum paper width	16 inches
Carriage motion	bidirectional
Paper feed	bidirectional, friction
Ribbon	cartridge type (Diablo)
Interface	Centronics parallel or RS-232-C serial
Dimensions	22.4 × 4.7 × 15.8 inches
Weight	30.8 pounds
Price (list)	$999



The Juki Model 6300 printer sports competitive print quality and a complete, helpful user manual.

MicroStandards

Editor: Robert G. Stewart/Stewart Research Enterprises/1658 Belvoir Drive/Los Altos, CA 94022

Eight long years . . . but success at last!

by Robert G. Stewart

The IEEE Standards Board put the official stamp of IEEE approval on six standards developed by the Computer Society's Technical Committee on Mini/Microcomputers during its March meeting in New York City. They are

- 694, Assembly Language for Microprocessors,
- 695, Relocatable Object Format (MUFOM),
- 754, Binary Floating-point Arithmetic,
- 755, High-Level-Language Extensions for Microprocessors,
- 855, Microprocessor Operating System Interface (MOSI), and
- 949, Media Independent Information Transfer (MIIT).

The first four of the above new standards were initiated as a result of the historic first meeting of the Computer Society's Microprocessor Standards Committee one nice summer evening in August 1977 in a restaurant in beautiful downtown Los Altos. Most of us present expected that two years at most would be required to develop the standards. Wisdom comes slowly. Let me tell you their stories.

694—Assembly Language. We quickly accepted the fact that instruction sets per se need to remain flexible and should not be standardized. Nevertheless, the commonality of instructions contained in the leading microprocessors suggested that it would be very helpful to define the mnemonics and operand sequence for the general instructions. That certainly would be a real boon to anyone using micros from different vendors. I consider it appalling that up till now the meaning of MOV A,B meant move (copy, in reality) the contents of register A into B on some processors, and the reverse on others. But progress is at hand! Instead of quicksand, there is now firm earth beneath our feet! IEEE 694 gives the move instruction a fixed meaning:

MOV A,B → move the contents of A, the source, into B, the destination.

And this took till the middle of the ninth decade of the twentieth century AD!!!

The source → destination sequence for MOV is also used for STO, the store instruction. However, in the case of LD, the load instruction, 694 specifies

LD A,B → load register A with the contents of register B.

The rule for selecting instruction names is that they are to be action verbs if at all possible. The existence of an IEEE standard for naming processor instructions should help reduce the number of unnecessary errors caused by the fact that users programming different microprocessors have had to deal with different names and sequences for the very same instruction. Some manufacturers have even copyrighted their mnemonics and have for a time declined to give permission to writers of assemblers to use them!

The advent of 16- and 32-bit micros delayed the development of 694 about two or three years, as the working group dealt with the much more sophisticated addressing schemes and complicated instructions they possess.

The first chairman of the 694 working group was Wayne Fischer. When the project started he was always affable. Wayne and I had a continuing difference of opinion on the correct pronunciation of the word "mnemonic." The first "m" is silent. Wayne pronounced the "e" like a "u," as in "true," whereas I pronounced the "e" like the "e" in "let." [Ed. note: Actually, Bob consistently pronounces the word "menomic," which prompted one wag to circulate a petition at an MSC meeting calling for an IEEE standard for spelling and pronouncing "mnemonic." Though the petition had wide support, no standard has yet resulted.] As the years went by, Wayne became involved with the P1014 VME bus standardization effort, so Geoffrey Baldwin took over 694 and developed the final standard.

Geoff's hobby is translating Chinese. Any of you who claim that Chinese ideographs were the principal source of ideas for 694 are probably completely mistaken. Another stalwart was the inimitable Tom Pittman. The little guy always had a dear friend in Tom. You should have seen the 4004 system which comprised Tom's working computer for many years. Rube Goldberg would have been pleased. (Sort of like my trusty MITS Altair, with which this article is being prepared.) Another regular participant in the early years was Andy Allison, later chairman of the P896 bus committee and now mayor of Los Altos Hills. We had the pleasure of Jean-Daniel Nicoud's presence while he was here on sabbatical from the Swiss Federal Institute of Technology. His professorial touch helped lead the working group through the complexities of extended addressing in 16- and 32-bit processors. Steve Savitzky also helped adapt the standard to the nuances of recent 32-bit processors.

Many others all over the world contributed by sending comments and suggestions to the working group after publication of drafts in *Computer* and *IEEE Micro*.. Sooooo . . . the challenge to microprocessor manufacturers and assembler writers is . . . use IEEE 694!

695—Microprocessor Universal Format for Object Modules (MUFOM). This standard defines a format for relocatable object code. The intent here is to enable modules generated by different assemblers and compilers to be used interchangeably. A number of different formats were studied by the working group, which was first chaired by Tom Pittman. Then Dave Gustavson of SLAC learned that CERN in Switzerland had an object format known as CUFOM which was being used very successfully to generate object modules on,

say, a large computer and download them to microprocessor-based systems over phone lines. CUFOM was able to perform the arithmetic calculations needed in the process of loading. Ian Willers of CERN visited UC Berkeley and joined the working group and helped develop the improvements and modifications that resulted in MUFOM. The load/relocate command was reworked considerably to improve usage of file space. A primitive library facility was provided. An iterate operator was added to allow a command to be repeated a number of times. This operator allows, for example, memory to be filled with an arbitrary number of floating-point operations, such as setting memory to Not-a-Number (NaN), which permits finding uninitialized variables. An error operator was added so a function can be calculated at link time, a function which, if true, results in a link-time error being indicated. This is needed because language processors may be cognizant of error conditions that could arise but that are computable only at link time when linking with other modules is established.

The MUFOM standard is independent of the width of the word of the target computer. It deals with "minimum addressable units" of the target machine. This permits the linker to generate code for a variety of targets rather than for just one. It is not designed for achieving a maximum packing density. A subsequent binary packing transformation can be performed on the module if higher code density is desired. The linker does not need to know the width of the target machine to do, for example, library searching. The width of the target becomes significant only when the loader loads the code into memory.

Active participants in the 695 working group included Geoff Baldwin, Rick James, Greg Marshall (the final chairman), Jean Montuelle, Tom Pittman, Steve Savitzky, and Ian Willers.

754—Floating-point Arithmetic. Who would ever guess that floating-point arithmetic can raise the passions that it does!

When we started 754, our motivation was to try to define some formats so the chip houses would not give us the Big Endian/Little Endian byte ordering treatment all over again in the case of arithmetic. Dick Delp, then of AMD, chaired the working group at first. In the fall of 1977, John Palmer of Intel presented a proposal to the working group. This proposal was the original basis of the 8087 coprocessor chip. John kept mentioning a Professor Kahan of the University of California at Berkeley, whom Intel had retained as a consultant in preparing algorithms for the arithmetic processor. So in the spring of 1978 I called "Vel" and invited him to join the working group. And did he ever! With Professor Kahan came Jerry Coonen, a PhD candidate, and Professor Harold Stone, then visiting at Berkeley. The three fashioned a sophisticated proposal of their own that went well beyond the original Intel proposal. This became known to the world as the "KCS proposal."

The KCS proposal offered a procedure known as "gradual underflow"to maximize the accuracy obtainable in a given number of bits. This is particularly important when a limited field width, such as single-precision, 32-bit arithmetic, is employed. Other features introduced were user control of rounding direction, which makes it possible to do interval arithmetic efficiently, and the NaN (Not a Number). Interval arithmetic calculates exact upper and lower bounds for the true answer and thus accounts for any loss of precision due to round-off and truncation errors accumulating in the calculation. Despite user pressure, and the original *raison d'être* for the committee, the working group dropped any attempt to specify the absolute memory storage or the communication transfer format for binary floating-point numbers.

Now that there is an approved IEEE standard for floating-point arithmetic, David Hough, formerly of Apple and now of Sun Microsystems, tells me that he has a software implementation of the standard as it existed in 1980. This software provides a comparison for results and exceptions and thus is useful for debugging new implementations. This program is called IEEE Calculator, is written entirely in Pascal, and computes results a bit at a time. Hence, it is too slow for any use other than code debugging of implementations of the standard. This program has been placed in the public domain, and is unsupported. It is available on Usenet, the UNIX network. Questions should be directed to sun!dgh.

755—High-Level-Language Extensions for Microprocessors. The motivation for this effort was the fact that microprocessors are frequently used in direct hardware applications. But groups standardizing the high-level languages, particularly those controlled by CBEMA's X3 committee, had bent over backward to make it impossible to gain direct access to hardware in a high-level language. Verily it was joyous witnessing the passionate advocacies of some HLL gurus that bits are simply not made to be twiddled. And that is that. The result is that programmers are frequently forced to use assembly language when they perhaps would prefer an HLL. In my opinion, the upsurge of the C language in recent years has been at least partly the result of its recognition of assembly language capabilities that permit better modeling and use of actual microprocessor architecture.

Frequently the functionality needed by 755 can be provided by standard subroutines and functions rather than by changes to the language standards themselves. The principal extensions defined by 755, and their function or subroutine names, are

- memory access—PEEK and POKE,
- port and discrete I/O—INP and call OUT,
- interrupts—ARM and DISARM,
- calling of assembly language routines—CALLER, GETREG, and PUTREG,
- bit manipulation—IAND, IOR, IXOR, and INOT,
- subfield access—GETFLD and PUTFLD, and
- address binding for ROM vs. RAM—ROM and RAM.

Bruce Ravenel originally chaired 755. He resigned to cochair the 770 Pascal working group. Rick James then provided the leadership for 755 until its adoption. Tom Pittman made substantial contributions based on his continuing efforts in designing software for micro systems. Others involved included Wayne Fischer, Dick Karpinski, Dennis Paull, and yours truly, Smiling Bob.

855—Microprocessor Operating Systems Interface (MOSI). This standard was initiated in 1981 to provide a specification by which application programs can interface with operating systems, and by which operating systems can interface with the underlying hardware.

Reader Interest Survey

Indicate your interest in this department by circling the appropriate number on the Reader Interest Card.

High 182 Medium 183 Low 184

Such a specification is highly desirable to users since it provides greater portability of software. Further, such an open systems architecture specification is advantageous to hardware manufacturers because it allows customized operating systems—needed, say, for real-time functions—to be used without requiring a rewrite of the application software when the operating system is changed. This clearly should be beneficial to vendors of different operating systems.

The operating system functions provided by MOSI include

- memory management,
- time management and scheduling,
- data transfer,
- process management,
- process synchronization and communication,
- interfacing with the computer's environment, and
- exception handling.

MOSI also specifies levels of compliance which denote levels of capability for different categories of applications.

Jack Cowan first chaired 855 and secured significant support from Intel in creating the initial drafts. Later, Don Jackson of Motorola became chairman and saw the drafts emerge in their final form. Language bindings for Pascal, Ada, and Cobol were included in the drafts, and Fortran and C bindings will be added.

Jim Mooney of West Virginia University is now working group chairman; he may be reached at (304) 293-3607. Additional active members of 855 were Marvin Conrad, Terry Hengl, Sam Kirk, Alan Marcum, and Richard Wirt.

Mooney is working on a CP/M implementation of MOSI. His students are working on a UNIX implementation. The CP/M implementation requires about 10K bytes of memory, with about 8K bytes used for file buffers and 2K bytes for interfacing to language compilers. Much of this memory partially substitutes for the usual run-time library of the operating system. Programs run as fast with MOSI because corresponding functions within the run-time library are deleted. Hence, a MOSI implementation does not suffer a speed penalty. The extra memory that it uses is compensated for by the extra capability MOSI provides.

949—Media Independent Information Transfer (MIIT). This effort was started in the spring of 1982, with Bob Davis as chairman. The standard provides a format for arbitrary files which allows communication or storage by any byte-stream-oriented method. Thus, it is applicable to a wide range of media—paper tape, floppy disks, punched cards, telephone lines, packet switched networks, and optical disks, for example. The address range is 2^{64} bits, which can cope with the address space of optical disks or be subdivided for smaller files. MIIT provides optional error detection or correction capabilities such as straight 16-bit checksum, 16-bit CRC according to the CCITT format, 32-bit CRC according to the IEEE 802 LAN standard, and 64-bit checksum, or it supports the user's own. The specific selection is indicated by appropriate bits in the description field. The standard provides a byte count, file type, file name, user-specified fields, date of creation, offset from zero, and an ownership field for indicating copyright information.

Tom Pittman developed a software implementation of the draft. Applications of MIIT have already been made to optical disk files, which benefit from the large address space.

Active participants in the working group included Nels Anderson, Bob Davis, Wayne Fischer, Tom Pittman, Steve Savitzky, and Mike Smolin.

I would like to thank Bob Davis, Dave Gustavson, Don Jackson, Jim Mooney, and Tom Pittman for assisting me in gathering the information for this column.

MicroReview *continued from p. 79*

to help a consumer take a serious look at his or her needs.

Unfortunately, the guide fails to take into account the latest technology in this field; and its catalogue of hardware and software products seems incomplete as well as dated. Some of the major competitors, such as Crosstalk 16 and the Hayes 1200B board modem for the IBM PC and compatibles, are completely missing.

These are major oversights, so while the introductory section is reasonably sound for the novice and the glossary is acceptable, the book, as a whole, is not acceptable as a "Buyer's Guide" to the subject under consideration.

Buyer's Guide to Computer Printers by Sanjiva K. Nath, Terry C. Silveria, and Edward Hogg (Tab Books, 1985).

Brought to you from the authors of the *Buyer's Guide to Modems and Communications Software,* this guide has the same inherent problems as its companion: it includes outdated information on printer technology and it lacks current product listings. I would not recommend these guides as good references for even the novice computer or data communications user.

The AT&T PC 6300 Made Easy by Martin D. Seyer and Leo J. Scanlon (Prentice-Hall, 1985).

This is a reworking of the authors' previous work, *The IBM PC Made Easy.* With the exception of a few minor errors, this is a reasonably good book for the novice user of both the AT&T PC and MS-DOS compatibles.

The following sections in this book are most useful:

- Computer Components
- Getting Started
- GW Basic
- Disk Operating System (MS-DOS)
- Appendix A: Formatting and Copying Disks
- Appendix E: Pinouts for the PC 6300 Ports

The Computer Components section gives a good though somewhat lengthy description of the PC 6300 hardware. The sections on GW Basic and MS-DOS provide a good feel for the underlying software for this machine, and Appendix A gives the new user a headstart on some very basic operations. This book could certainly give the new user another source for learning the system, apart from the PC 6300 standard documentation.

Thanks to my readers for their comments on this column. If there is any specific hardware, software, or book you would like reviewed, please fill out the Reader Interest Card at the back of this magazine. I mean it when I say, "Keep those cards and letters coming!"

Reader Interest Survey

Indicate your interest in this department by circling the appropriate number on the Reader Interest Card.

High 179 Medium 180 Low 181

MicroNews

What is an engineering workstation?

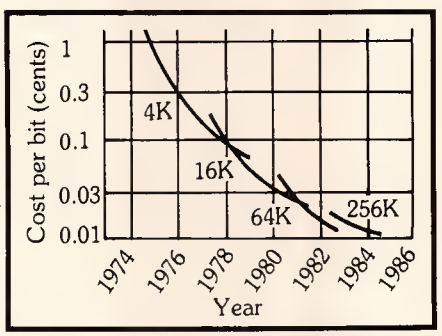
by Richard Landry, Managing Editor

"An engineering workstation is not a VT 220," said Masscomp representative Rathan Dar in a presentation delivered at the April IEEE Microprocessor Forum in Atlantic City, New Jersey. Although virtually all participants in the session could agree with Dar on this point, their differences on other issues underscored both the diversity and the potentially disruptive dynamism of the present workstation market.

"Figures of merit." What should an engineering workstation be and what should it be expected to do? According to Dar, it all depends upon the productivity "figure of merit" that a company hopes to achieve through the use of workstations. "If you want to cut down your development cycle from a year to four months, then obviously your requirements are different than if you wanted to cut it down from a year to nine months. Engineering workstations provide performance, but at a cost."

For Dar, a workstation capable of providing top performance must exhibit certain features: it should be a 32-bit system with a virtual-memory environment, offering sophisticated graphics and networking capability. "You can add a coprocessor to a 16-bit system," he noted, "but it's not the same as a 32-bit system with a commonly shared database." More specifically, Dar said that he would expect processor performance in the range of 2.4 to 4.0 MIPS, with floating-point operations at the speed of about 1.2M Whetstones per second. Further, he would expect high-performance graphics, supporting the GKS/GCM/GCI graphics interface standards, at a competitive price—"One-half the cost of a workstation is in the graphics system, including monitor," he noted—with a mouse or tablet pointing device. Finally, he would demand a networking capability to move data between systems—to an array processor, for example, for solid modeling functions that require powerful number-crunching capability.

Dar found support for his position, in part at least, from several other Forum speakers. Hewlett-Packard's Rick Cioto echoed the sentiment that the future belonged to systems that offered a networked environment, although his emphasis lay more in its impact on cost-per-seat than upon sheer performance. Linking workstations, on one end, to low-cost PCs and, on the other end, to a high-performance database manager might offer the best overall productivity gains, he said. David Burns of Daisy Systems agreed that the company that offered a hierarchy of design capabilities at a range of prices would be in the best position to make a case for the engineering workstation as a valuable productivity tool.



Cost per bit of a dynamic RAM. Source: Strategic, Incorporated.

Computers at $200 per pound. What places this or any purchase strategy on shaky ground is the continuing increase in performance and accompanying decrease in price of all computers, across the board, as Apollo's David Nelson testified. Whimsically observing that computer prices have remained stable at about $200 per pound of hardware, he noted that performance has leaped by an order of ten every seven years, so that "the characteristics of computation in one environment continually reappear in the next smaller environment," he said. Low-end workstations quickly drop out of sight under these circumstances, while the performance of expensive, high-end workstations migrates to the next lower rung on the design-capability ladder. Distributed processing now permits engineers to spread the cost of computation-intensive functions such as ray tracing over a networked workstation environment, he noted for example, while the next step—ray-tracing in real time—would still take "about 100 Crays" at present, he said. In short, performance-per-pound improvements will place increasing pressure on both ends of the design hierarchy.

DRAMs at one-hundredth of a cent per bit. This forecast, exciting for users but frightening to workstation manufacturers and procurement departments alike, is highlighted in several recent industry reports. One such report, "Technology Trends in High-Performance Workstations," by Cupertino, California-based Strategic, Incorporated, baldly states that the research firm "does not see any end to the continued improvements in processor technology" and, by inference, to price reductions. Whereas, for example, "the cost per bit of a dynamic RAM . . . was measured in cents during the 1970s," the report says, "it is now being measured in hundredths of a cent" (see accompanying graph). Concurrently, processor output is rising so dramatically that, by 1991, the firm expects that performance will be in the range of 10 to 20 MIPS. Storage is expected to drop from the current price of $60 per Mbyte to $25 per Mbyte by 1989, and the cost of a low-end workstation itself (defined as a 32-bit graphics-oriented system) will drop to $10,000 by the end of this year, with continuing drops of five to eight percent per year.

All this seemingly good news may suggest that the great industry shakeout is yet to come; and along with this shakeout will follow yet another redefinition of the term, "engineering workstation."

Micromouse squeaks at Microprocessor Forum

by Richard Landry, Managing Editor

It did not exactly qualify as an infestation, but the single entrant to last April's Computer Society Micromouse contest, held at the IEEE Microprocessor Forum in Atlantic City, New Jersey, more than made up for the lack of competition with a run that a-mazed onlookers and upstaged an exhibition-class electronic rodent from Japan.

The mouse, designed by a California State University-Los Angeles undergraduate team led by electrical engineering professor Jack Levine, negotiated an official Japanese Micromouse Association maze in less than 15 minutes, while Mapi, its Japanese counterpart, was frustrated by continuing snags at the maze's sharp turns.

What was the secret of the Cal State-LA team's success? According to Levine, devising an algorithm to get the mouse to find the center of the maze from an entrance point at its perimeter was just about the easiest part. Most of the team's effort, he said, went into programming the microprocessor that drove the tiny robot and in putting together a smooth-running body.

As Mapi's frustration testified, getting the mechanics of the mouse to mesh with its brain was no unimportant matter. Dressed like a police officer with hat, club, and built-in siren, Mapi (a play on the Japanese word "Mopo," or cop) was intended to search the maze for its center where there lay in wait a balloon imprinted with the image of a "cat burglar," burst the balloon, and then trace the shortest route back to the starting point. Although its crime-stopping efforts met with success once, Mapi could not duplicate them, and it truly became flummoxed when it attempted to negotiate a maze rearranged by members of the audience.

Levine's team will get a chance to repeat its achievement when it travels this August to Japan to compete in the 1985 World Micromouse Contest against entrants from Japan, Korea, and Europe. There the competition will be stiff: the Japan Micromouse Association boasts about 1000 members and between 50,000-60,000 readers of its magazine. Of the 70 entries that ran in the 1983 contest, only 20 completed the maze in 15 minutes or under. Under these circumstances, noted Mapi, anyone who takes the bait is a big cheese.

For more information on the micromouse, contact Micromouse Contest, IEEE Computer Society, 1730 Massachusetts Avenue, NW, Washington, DC 20036-1903.



This California State University-Los Angeles team travels to Japan this August to compete in the 1985 World Micromouse Contest (from left): Roland Torres, Vivian Voskian, Baxter Cheung, Araks Matevosyan, and Joseph Chiu.

DRAM stores 1M bits

Fujitsu Limited has developed a dynamic random-access memory (DRAM) that can store 1M bits of data. Described this year at the International Solid State Circuits Conference, the Fujitsu DRAM is designed with NMOS technology and contains a new memory-cell structure that is smaller and more resistant to failures induced by alpha particles.

The new memory-cell structure, called a three-dimensional stacked capacitor cell, offers data-storage capacitance as high as 55 picofarads in a cell area of 26.5 square microns. The high capacitance is due to use of the curvature and sidewall of the second layer of polysilicon, unique to the three-dimensional structure.

The three-dimensional structure is said to outperform traditional structures such as the trench capacitor or Hi-C structures by a large margin. For example, Fujitsu says that the new structure is about 1000 times more immune to soft errors caused by alpha particles, compared with the trench capacitor cell. As a result, Fujitsu engineers expect to use the new cell structure for devices of 4M-bit density and beyond.

The DRAM's chip layout divides the device's memory-cell array into two blocks, with peripheral circuitry and some input/output pads located in the center of the chip. This approach is said to result in a significant reduction of inner leadframe length. The DRAM was also designed to conservative 1.4-micron design rules, to make the device practical and producible, Fujitsu says. The three-dimensional cell is an extension of the three-layer polysilicon memory cell currently used by Fujitsu to mass produce a 256K-bit DRAM.

The DRAM is organized as 1,048,576 1-bit words. Access times are 90 nanoseconds (t/rac/) and 45 ns (t/cac/). Power consumption is 350 milliwatts operating and 15 mW in standby, from a single 5-Volt power supply. Chip area is 12.32 by 4.44 mm, for a total of 54.7 square millimeters. Cell area is 8.4 by 3.15 microns.

The Fujitsu DRAM is scheduled for volume production in late 1986. For more information, contact Fujitsu Microelectronics, 3320 Scott Boulevard, Santa Clara, CA 95051; (408) 727-1700.

IEEE Micro authors win Thompson Memorial Prize

Douglas MacGregor and David Mothersole have been granted the 1985 Browder J. Thompson Memorial Prize Award for an article which appeared in the June 1983 issue of *IEEE Micro*.

MacGregor and Mothersole won the award for their article, "Virtual Memory and the MC68010." Most recently, they coauthored, with Bill Moyer, the article, "The Motorola MC68020," which appeared in the August 1984 issue of *IEEE Micro*.

The Thompson Memorial Prize is awarded by the IEEE to an author or authors under the age of 30 at the time of submission of the most outstanding paper published in any of the IEEE publications. The 1986 Thompson Prize will go to the most outstanding author or authors in this category whose work was published in 1984.

Mothersole was the project leader at Motorola for the MC68020 design and is currently researching new VLSI computer architectures. Previous to this position, he spent six years working on the design of the M68000 family of processors. His interests include computer networking. He received his BS and MS degrees from the University of Texas at Austin.



Browder J. Thompson Memorial Prize winners David Mothersole (left) and Douglas MacGregor (right) in the photo that appeared with their June 1983 article, "Virtual Memory and the MC68010."

MacGregor designed the control structures and wrote the microcode for the MC68010 and MC68020. He received a BA degree in history and Asian studies, and an MS degree in computer science from the University of Illinois. Fluent in Japanese, MacGregor is currently a doctoral candidate at the University of Kyoto.

SDIC announces call for sessions

Organizers of the Systems Design & Integration Conference (formerly Mini/Micro West), scheduled for January 28-30, 1986 in San Francisco, California, have announced a call for session chairpersons. SDIC bills itself as "the only regional conference focussing exclusively on computer system design and integration." Individuals interested in participating in the conference as session organizers should contact Dale Litherland, SDIC, 8110 Airport Boulevard, Los Angeles, CA 90045; (213) 772-2965.

Reader Interest Survey

Indicate your interest in this department by circling the appropriate number on the Reader Interest Card.

High 185 Medium 186 Low 187

Landry named *IEEE Micro* Managing Editor



***IEEE Micro* Managing Editor Richard Landry.**

IEEE Micro Editor and Publisher True Seaborn has announced the appointment of Richard Landry as managing editor of the magazine.

Landry was appointed last February after serving as assistant editor for the Computer Society publications *IEEE Software* and *IEEE Design & Test*. Previous to these positions, he taught English literature at Scripps College, Claremont, California, and at Stanford University. He has also been a freelance writer and a newspaper editor.

Landry received an AB degree from the University of Notre Dame and an MA degree from Stanford University, where he is currently a PhD candidate in English and American literature. He was a Notre Dame Scholar and a Stanford Fellow.

Editor: Victor P. Nelson

New Products

Static RAM offers on-chip diagnostics

The Am9151, a 1K × 4-bit static RAM with on-chip registers, is now available from Advanced Micro devices.

According to the company, the Am9151's registers provide pipelining, initialization, and diagnostic capabilities. The RAM is manufactured using NMOS techniques to achieve a 40-ns cycle time. Diagnostic capability is provided by the AMD Serial Shadow Register. The SSR is said to detect system- or board-level malfunctions due to hardware or firmware that use serial-scan diagnostics. It can also be used to load microcode serially into the Am9151 memory array; the array in turn is used as a writable control store in pipelined, microprogrammed system applications.

The Am9151 is available in a 24-pin ceramic dual-in-line package in both commercial and military versions. The commercial, 40-ns version is priced at $25 each in 100-unit quantities; the military, 50-ns version is priced at $45 each in 100-unit quantities.

Advanced Micro Devices is located at 901 Thompson Place, Sunnyvale, CA 94086; (408) 732-2400.

Reader Service Number 6

Octal serial board runs at 10 MHz

SBE Inc. has introduced the M68COM intelligent octal serial board as an addition to its ModulasTen line of Multibus-compatible products.

The board provides eight independent full-duplex channels. Running on the 68000 or 68010 microprocessor at 10 Mhz with no wait states for memory reads or writes, the M68COM includes an optional 10-MHz, four-channel DMA controller for high-speed data links.

Four multiprotocol serial communications controllers on the board allow each of the eight channels to be programmed independently to run in asynchronous, byte-synchronous, and bit-synchronous modes. The board supports X.25, SDLC, HDLC, and Bisync protocols. Each of the eight ports can be independently configured as RS-232, RS422, RS-423, fiber-optic, or direct-connect channels, and custom interfaces are also available.

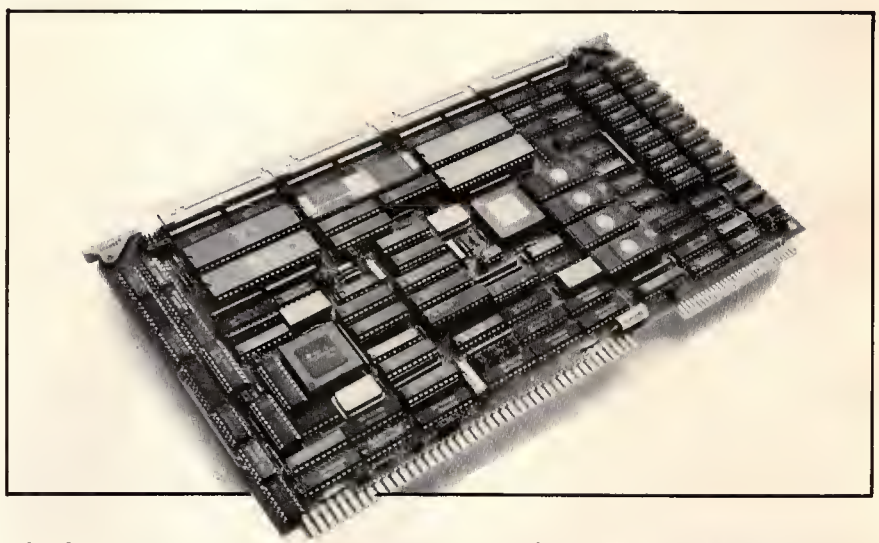
For on-board RAM, the M68COM offers either 128K or 512K bytes of dual-ported, hardware-refreshed dynamic RAM with parity protection. In addition, four on-board EPROM sockets accommodate up to 256K bytes of EPROM.

SBE notes that the M68COM can be used as a data concentrator, high-speed communication link, main master in a Multibus system, or I/O support processor, especially in UNIX-based systems. It can be ordered with the UNIX-like Regulus operating system, which includes the Probug debugger.

Each channel on the M68COM has three interrupt sources, each of which can be enabled under program control. The board also includes a mailbox-interrupt feature, which allows another Multibus master to generate a local interrupt on the M68COM. A 68230 Parallel Interface/Timer chip provides an on-board timer that generates periodic system interrupts to the M68COM. The board supports programmable baud rates from less than 5 bps to 38.4K bps, and external baud rates from 0 to 880K bps.

The M68COM is available with a 10-MHz 68000 MPU, 128K bytes of dynamic RAM, Probug, and a four-channel DMA controller for $1595 in quantities of 100. SBE Inc. is located at 2400 Bisso Lane, Concord, bA 94520; (800)221-6458; (415) 680-7722 in California.

Reader Service Number 7



The SBE M68COM communications board can be configured as an I/O support processor in UNIX-based systems, according to SBE officials.

Supermicro accommodates up to 30 users

Altos Computer Systems has announced the Altos 3068 multiuser supermicro, which is said to accommodate up to 30 users in different system configurations.

Designed for the OEM marketplace, the 3068 uses the UNIX System V operating system and the Motorola 68020 microprocessor. It can be networked with other Altos multiuser systems via the Altos WorkNet local-area network and with mainframes via 3270 Bisync, SNA, X.25, and 3780 communication options. With the Altos PC Path attached to Worknet, the 3068 can act as a file server and communications gateway for personal computers.

The 3068 base configuration supports up to 10 users with 1M bytes of RAM, a 20M-byte hard disk, and a 1.2M-byte floppy disk. RAM can be expanded to 16M bytes in 1M-, 2M-, or 4M-byte increments. Hard disks can be upgraded to 240M bytes (unformatted) in 20M-, 60M-, or 80M-byte increments. A streaming magnetic tape unit, with up to 60M bytes of backup storage, is also available.

The modular design includes eight board slots, four of which can be custom-configured. The UNIX System V operating system supports demand-paged virtual memory with a 1K-byte page size. The 3068 also features a set of horizontal applications software, including AOE, Altos' office automation package, and a set of test and diagnostic options. Local diagnosis is possible, as is remote analysis via modem to the factory or authorized OEM service center.

The Altos 3068 is available beginning in July, with a base price of $7000 in OEM quantities. Altos Computer Systems is located at 2641 Orchard Parkway, San Jose, CA 95234 (408) 946-6700.

Reader Service Number 8

Interactive stat package designed for Macintosh

Brainpower, Inc. has announced StatView, an interactive statistics program for the Macintosh computer.

A technical utility designed for data analysis, StatView takes advantage of the Macintosh computer's windows, pull-down menus, and mouse. According to the company, most statistics packages do not permit simultaneous viewing and analysis of data. However, while keeping data in front of the user, StatView permits use of the mouse to select data for analysis and to choose type of analysis from a pull-down menu. Results appear dynamically in another window as changes are made in the data. A variety of graphic representations of data or results are immediately available in a window alongside data on the screen. These include tables or bar, line, and pie charts, and scattergrams.

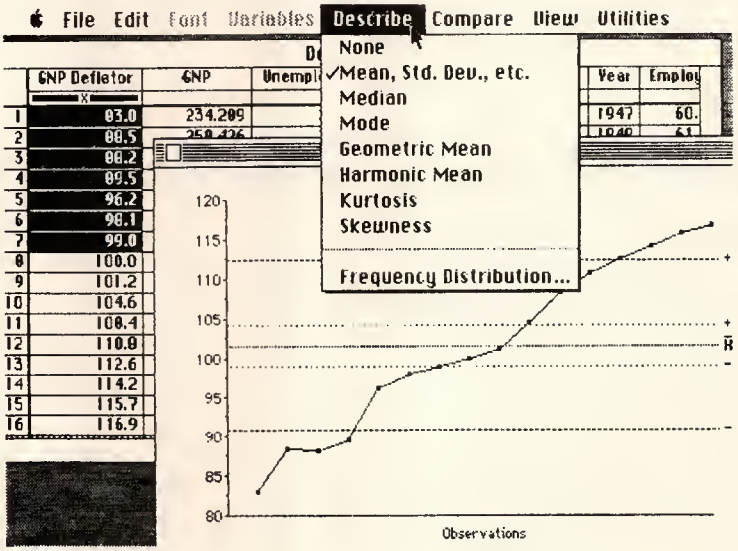
StatView is designed for use on all Macintosh formats including 128K and 512K Macintoshes and the 4M-byte Lisa equipped with Mac Works, the Macintosh emulation disk.

Brainpower notes that many types of statistical evaluation are possible with StatView, including descriptive statistics, comparative statistics, and non-parametric tests. StatView's descriptive capability includes harmonic and geometric mean, standard deviation, standard error, variance, median, mode, frequency distribution, and coefficient of kurtosis and skewness.

Comparative statistics include paired and unpaired t-tests, Searman rank-order correlation coefficient, ANOVA (Analysis of Variants) with percent variation explained, and multiple/simple multiple/and polynomial regression. Nonparametric tests include chi-square, Wilcoxon signed rank, Kruskal-Wallis, Friedman, Kolmogorov Smirnov, Mann-Whitney-U, and Wald-Wolfowitz runs.

StatView is scheduled to begin shipping July 15. The StatView program will carry a suggested retail price of $199.95, including manual and a copy of the book *Statistics,* by Vicki Sharp. Special discounts will be available for high schools, colleges, and other educational institutions. Brainpower, Inc. is located at 24009 Ventura Blvd., Calabasas, CA 91302; (212) 986-6668.

Reader Service Number 9



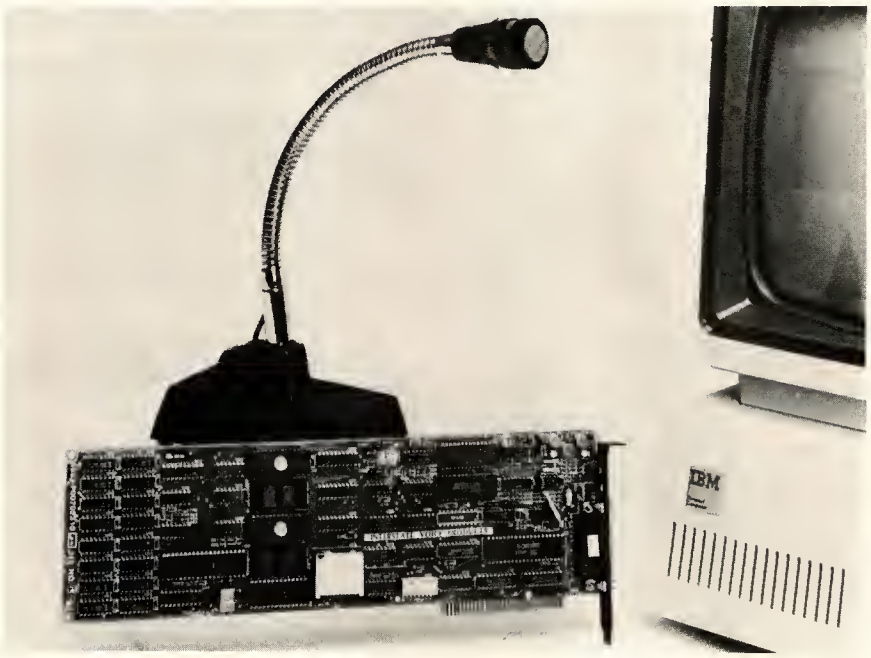
StatView is an interactive statistical analysis package designed to take advantage of the Apple Macintosh's graphics capabilities.

For more information, circle the appropriate Reader Service Number on the Reader Service Card at the back of the magazine.

Voice board for PC allows more natural speech style

Interstate Voice Products has announced a voice recognition board for the IBM PC and compatibles that offers connected speech recognition capability. This is said to eliminate the need for pauses between utterances, allowing users to input data in a more natural speaking style.

Designated the Vocalite CSRB240, the voice recognition board allows users to structure a vocabulary of 15 to 20 words, including digits and control words, which may be uttered in a connected fashion. The 15-to-20 command/digit connected speech vocabulary may also be used in conjunction with a 240-word vocabulary of discretely uttered words. When used in this manner, key vocabulary commands may be defined by the user to evoke the connected speech algorithms embodied in the firmware of the product. For example, the key word "numbers" may be assigned to evoke dynamic programming algorithms designed to process a connected string of digits.



Interstate Voice Products' VocaLink CSRB240 voice recognition board for the IBM PC and compatibles incorporates a connected speech recognition feature which allows users to input data in a natural speaking style.

The new product evolves from the company's SRB board for the IBM PC, which performs "speaker-dependent" voice recognition. By taking advantage of the capabilities of the high-speed Intel 80186 microprocessor and 128K bytes of RAM on the CSRB240, Interstate Voice Products has implemented a new version of dynamic programming in the voice recognition firmware. The new dynamic programming algorithms are intended to eliminate the need for discrete pauses between utterances.

The single unit price for the CSRB240 is $1650 retail. Interstate Voice Products is located at 1849 Sequoia Ave., Orange, CA 92668; (714) 932-9010.

Reader Service Number 10

Development system for TMS320-based DSP

Whitman Engineering has announced the TMS320 Product Development System, which provides design engineers with an integrated software approach to creating TMS320 chip-based products. The package includes Whitman Engineering's Modular Access Design, Computer-Aided Processing (MADCAP) software's 12 main features, and five preprogrammed links between Texas Instruments' TMS320 Assembler/Linker/Simulator, Atlanta Signal Processor's Digital Filter Design Package, and a standard MS-DOS text editor. There is also a main menu selection for a data acquisition interface. Three user-designated module links are provided to interface packages such as schematic design, circuit board layout or prototype, and production hardware.

MADCAP features Practical Transform Lengths (PTL), which provides 94 FFT/IFFT lengths in addition to ten standard power-of-two lengths between 2 and 1024. Both forward and inverse transforms are computed with full 32-bit floating-point arithmetic. The user-friendly, single-stroke entry screens make MADCAP immediately productive for nonprogramming users, Whitman says.

According to the company, the user can write system specifications, develop a software model, write, assemble and simulate TMS320 code, test the prototype hardware, and test the production product. All steps utilize MADCAP's automated input signal generation, output signal display and frequency domain analysis for well-controlled, efficient and economical product development. These capabilities are said to convert an IBM PC, XT or TI PC into a development station for TMS320-based products as well as a full signal processing laboratory.

The price is $1795. Delivery is 2 weeks ARO. A MADCAP Demo Disk is available for $20 with the price deductible from purchase of the software and refundable in 20 days. Whitman Engineering is located at 520 S. Maitland Ave., Maitland, FL 32751; (305) 628-4516.

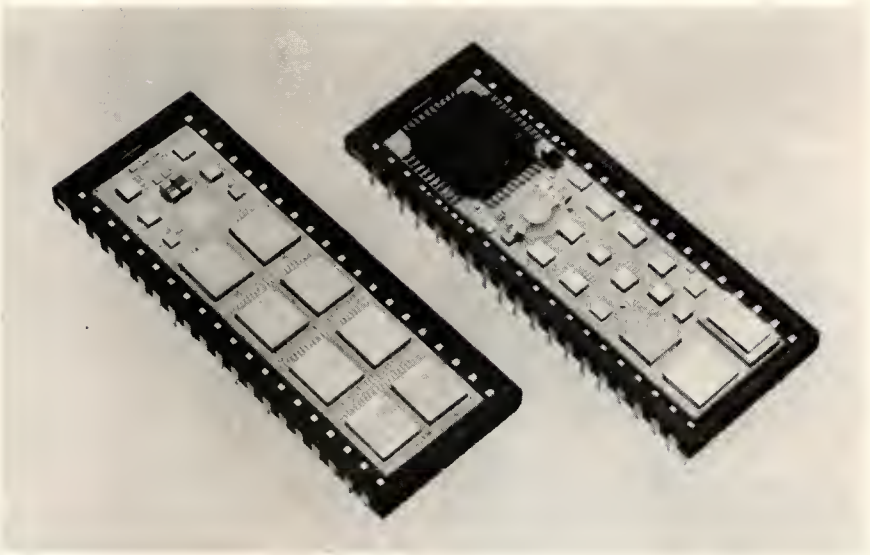
Reader Service Number 11

8-bit hybrid system fits standard 40-pin package

White Technology has announced the Model DHC8-P85 hybrid 8-bit microprocessor system for operation in hostile environments of between –55 and +200 degrees Celsius.

Fully CMOS,the stand-alone system includes an 8085 microprocessor, crystalclock oscillator, buffering, hardware UART, 2K static RAM, and 2KPROM, housed in a 2.0 × 0.8-inch standard 40-pin dual-in-linepackage. The input and output of a hardware Programmable Asynchronous Communications Interface (PACI) are used. This interface provides programmable baud rates from 50 to 9600 bps to permit interfaces to most devices that use RS-232 communications signal formats. Operating voltage for the module is +5Vdc and operating current is 35mA (4.0mA in hibernate mode). According to the company, the system ran over 10 hours on a Pt68 photo battery in laboratory tests.

The device is supplied with an internal monitor to bring the system up when power is first applied. This monitor can be turned off, and users' software can be written to any 2K PROM. The monitor can also be reprogrammed if desired, at 25 degrees Celsius. The company notes that, for production usage, it will copy and install users' code so that the system will come up running with the customers' software.



The White Technology DHC8-P85 hybrid 8-bit microprocessor is designed in CMOS for operation in hostile environments.

For applications requiring more memory, the DHC8-P85 system can be expanded to 64K in 16K increments, using White Technology's model DHC8-M16 16K-byte static CMOS RAM memory module. This module is also housed in a 40-pin DIP and operates from a +5Vdc power supply. The module features byte-wide organization and on-board address latches and decoders.

The DHC8-P85 microprocessor is priced at $2400 and the DHC8-M16 memory is priced at $1600 for unit quantities. White Technology is located at 4246 East Wood Street, Phoenix, AZ 85040; (602) 437-1520.

Reader Service Number 12

System transmits voice, text, graphics by modem

Chorus Data Systems has announced PhotoMail, which allows PC users to capture images with a standard video camera or VCR and transmit them to a remote PC via telephone lines. Still-frame pictures are sent at a resolution of up to 640 × 400 × 16 colors or levels of gray. PhotoMail operates with IBM PC, XT, AT and compatible computers.

PhotoMail is icon-driven and supports voice communications so users can converse while an image is being displayed. Each party can simultaneously point to a specific area of the picture by mouse or keyboard. Images can also be annotated with text. In addition to video images, PhotoMail includes a screen "grab" routine. This lets the user save and transmit IBM 320 × 200 × 4 color graphic displays. Screen displays generated by popular software packages, such as Lotus 1-2-3, can be sent to a remote location for display and printout.

The communications icon supports the Hayes Smartmodem 1200B and compatible modems as well as popular 2400 baud modems. Full screen images can be compressed and transmitted in under one minute. Once an image is transmitted, it can be saved on a disk or printed using the IBM graphics printer, Epson graphics printer, or the HP ThinkJet. The company says that it also plans to provide laser printer support and support for 9600 baud modems. In addition to direct PC-to-PC communication with pictures, PhotoMail also can format images to be used with electronic mail services such as MCI Mail, SOURCEMAIL, and other private mail systems.

The PhotoMail communications kit is priced at $2495 and includes the PC-EYE 1000 video digitizer, graphics display card, ScreenMaster, mouse, and PhotoMail software. PhotoMail software alone is $795. Delivery is four to six weeks ARO. Chorus Data Systems is located at 6 Continental Blvd., PO Box 370, Merrimack, NH 03054; (603) 424-2900.

Reader Service Number 13

For more information, circle the appropriate Reader Service Number on the Reader Service Card at the back of the magazine.

Dual-mode graphics system operates on IBM PC

TAT Graphics Group has announced the Sextant graphics display systems for the IBM PC, which are said to support low-resolution business graphics and high-resolution CAD/CAM design software on the same monitor.



The TAT Sextant graphics display system supports either low- resolution business graphics or high-resolution CAD/CAM software on the same monitor, the company says.

The Sextant systems include a single-board controller and a color monitor, available in either 14- or 18-inch display. To obtain dual-mode operation, the user requires a Sextant color monitor and an IBM color adaptor card. The TAT Galaxy G640, G800, and Gt5024 controllers are based on the NEC 7220 graphics processor. The systems offer a display of 16 colors from a palette of 4096, with standard 640 × 480 pixels of resolution and optional resolution of 1024 × 768 pixels. Programmable display parameters include pixels/line; lines/frame; sync and blank times; and interlaced/noninterlaced operation up to 40 MHz.

Prices for the Sextant graphics display systems are as follows: $1850 for the G640 or G800 controller; $2400 for the Gt5024 controller; $1700 for the Sextant 14-inch monitor; and $2900 for the 19-inch monitor. TAT Graphics Group is located at 1270 Lawrence Station Road, Bldg. E, Sunnyvale, CA 94089; (408) 734-2202.

Reader Service Number 14

Processor convertor chip mates 80286 to 8088

Edsun Laboratories has announced the EL286-88 application-specific VLSI integrated circuit, with the capability of converting the signals of the 16-bit 80286 processor into the equivalent signals of the 8-bit 8088 processor.

The EL286-88 translates data width, control signals, and circuit timing. To the 80286, the EL286-88 appears as 16-bit memory and/or peripheral devices operating at the 80286 clock rate; to the 8088, the EL286-88 appears as the 8-bit processor operating at its own clock rate. The two processor clocks can operate completely asynchronously, so that the 80286 can run at 8MHz yet address peripherals designed for 4.77 MHz data rates. Any 16-bit transfers requested of 8-bit devices by the 80286 are converted into multiple 8-bit transfers by the EL286-88. All translation is performed in hardware, without need for software modification. The EL286-88 also includes an on-chip clock generator and 16-bit bus controller.

Available in a 68-pin carrier and implemented in CMOS, the EL286-88 is priced at $44 in quantities. Edsun Laboratories is located at 7 Sears Road, Wayland, MA 01778; (617) 358-5667.

Reader Service Number 15

Micro development system eliminates emulator pod

Compu-Mech, Inc. has announced what it calls "the first truly universal microprocessor development system." Entitled the Universal Development Laboratory, or UDL, the system is said to transform a PDP-11, VAX, or personal computer into a development system that includes emulator, analyzer, and EPROM programmer.

The UDL eliminates the emulator pod by using target-independent hardware for emulation and state analysis. The personality of the target micro is defined in software, allowing the user to retool for less than conventional costs, the company says.



The Compu-Mech Universal Development Laboratory offers target-independent hardware for emulation and state analysis.

The UDL is controlled from a host computer over a serial link by means of a software package. The package provides a 48-channel bus state analyzer, four-step sequential triggering, selective tracing, breakpoint disassembly, memory, and register-port modification. Software support is currently available for over 30 processors, including the 8086, 8048, 8051, and 6800 families.

The UDL accepts output from cross-assemblers in Intel Hex or binary format, and comes supplied with manual, software, and necessary cables. The company offers a 15-day trial period. Compu-Mech is located at 5242 Angola Road, Suite 75, Toledo, OH 43615; (419) 535-6702.

Reader Service Number 16

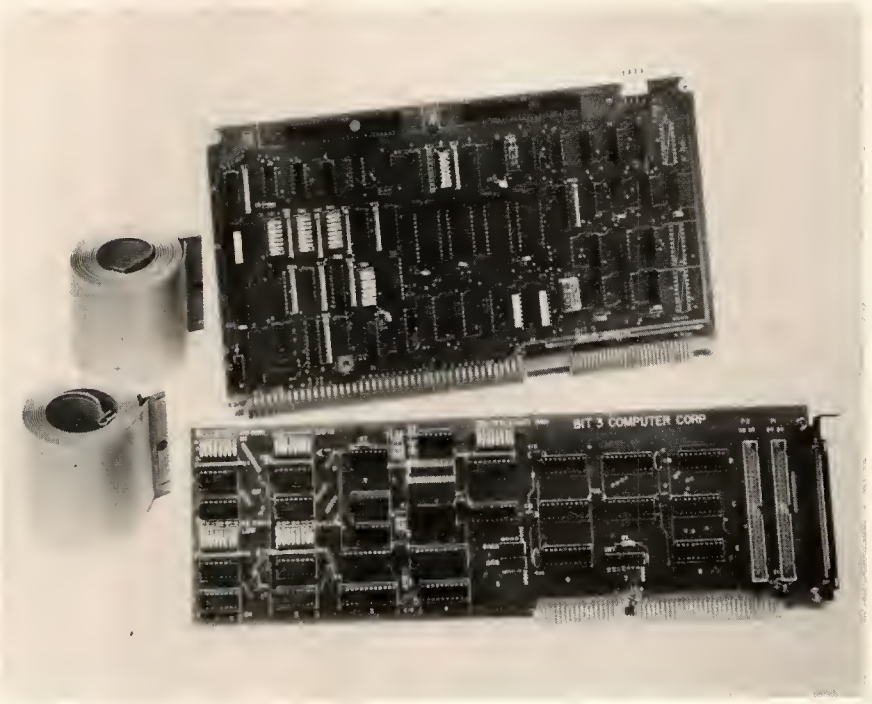
Multibus Adaptor lets PC/AT act as bus master

Bit 3 Computer Corporation has announced the Multibus Adaptor, which allows the IBM PC/AT to act as a Multibus system bus master or coprocessor.

The Adaptor consists of two interconnected cards, one of which fits in the PC/AT while the other occupies a card slot in the Multibus card cage. So configured, the PC/AT and Multibus system can communicate in three ways: up to 4M bytes of Multibus memory can appear as PC/AT memory; an 8K-byte static RAM on board the Adaptor, optionally expandable to 32K, can act as a dual-port concurrent access memory that can be accessed by both the PC/AT and the Multibus at the same time; and the Multibus I/O can be directly addressed as PC/AT I/O. The Adaptor features a data-transfer bus width of an 8-bit byte or 16-bit word for both memory and I/O cycles.

The Bit 3 Multibus Adaptor is priced at $1175. Bit 3 is located at 8120 Penn Avenue South, Minneapolis, MN 55431; (612) 881-6955.

Reader Service Number 17



The Bit 3 Multibus Adaptor is designed to link the IBM PC/AT with the IEEE 796 Standard Multibus.

Optical disc link allows PCs to read and write

Geac Computer Corporation has announced Gig-Attach, a product that allows personal computers to store and retrieve data from the Geac Gigadisc optical disc. The system consists of a board and software necessary to permit an IBM PC or compatible to both read and write on Gigadisc under MS-DOS. Geac says that only one of the attachments is necessary to allow a network of PCs access to Gigadisc storage. A UNIX version of Gig-Attach is presently under development. The Gigadisc can store up to 2G bytes of data.

Gig-Attach is priced at $1195 Can. Geac Computer Corporation is located at 350 Steelcase Road West, Markham, Ontario, L3R 1B3, Canada; (416) 475-0525.

Reader Service Number 18

Backup tape system designed for networked PCs

Advanced Digital Information Corporation has announced a high-capacity, file-addressable tape cartridge backup system for networked PCs. Called the Data Library, the tape system has a formatted capacity of 134M bytes, 32 tracks, and more than 140,000 addressable blocks. Capable of storing over 30,000 DOS files, the file-addressable Data Library emulates a hard disk, utilizing MS-DOS commands. Archive and Restore software is included with the system with features that allow unattended backup.

According to the company, data integrity is one of the key features of the new Data Library. Tapes are internally certified and verified. Cartridges are guaranteed to be interchangeable from system to system, and an automatic error detection/correction feature is said to reduce error rate to less than one in 100 billion bits.

ADIC accepts turn-key responsibility for the total tape drive backup system, including interface, tape drive, cartridge and software. Field maintenance is not required. The Data Library carries a one-year warranty. Operating on ac line voltages from 100V to 240V ac, and 47 to 60 Hz, the Data Library is UL listed and CSA, FCC, and VDE approved.

ADIC's Data Library system interfaces to Novell "Netware" software and is priced at $4495. Complete information on ADIC's Data Library tape cartridge backup system is available from ADIC, PO Box 2996, Redmond, WA 98013; (800) 638-0818 or (206) 881-8004.

Reader Service Number 19

For more information, circle the appropriate Reader Service Number on the Reader Service Card at the back of the magazine.



Telebyte Technology's Packetape archival storage system for the IBM PC/XT and AT features an error-checking-and-correction recording mechanism.

Cartridge backup system stores up to 67M bytes

Telebyte Technology, Inc. has announced the availability of Packetape, an IBM PC AT/XT archival storage and backup subsystem that utilizes cartridge tape.

The 67M-byte, 0.25-inch data cartridge subsystem incorporates a half-slot interface for use with the IBM PC AX/XT and compatible computers. It is based upon the 3M Company's HCD-75 drive and the DC600HC block-addressable cartridge. The Packetape formatter is said to enhance the 3M recording format with on-the-fly error detection and correction. This tape performance management is transparent to the user.

The Packetape system appears to the IBM PC AT/XT as four 16.75M-byte floppy disk drives. All files are randomly accessible on-line through the use of standard PC DOS commands. A 70K-byte dynamic cache buffer allows fast access to current files. Data transfer to and from tape is 35K bytes per second. Typical access time to any file is under 15 seconds. Track format features serpentine recording, which is designed to eliminate delays caused by rewinds.

The multipurpose subsystem is intended for backup of Winchester disk drives and for auxiliary on-line and removable archival storage. A single cartridge can be used for active backup and additional data and program files. Data logging and interchange between sites and applications requiring large volumes of readily accessible data are other uses. The interchangeability of cartridges provides a method of distributing large databases and software programs.

The Packetape IBM PC AT/XT subsystem contains a single-track, single-gap precision stepping head with straddle erase, which provides data integrity and single block overwrite.

Corrections of dropouts up to 512 bytes is permitted by an error-checking-and-correction recording format. A "Data Certify" command maps out any block with a single dropout, even when the data is correctable. This is said to statistically assure error-free operation for 100 or more complete reads or writes of a cartridge's full capacity. Read/Write/Verify is performed over 600 feet in 80 seconds at 60 ips.

Upon insertion of a cartridge, tape is wound from end to end in order to provide retensioning, automatic self-test, and head-to-tape edge alignment for assuring data interchange. The unit also provides automatic power-up test of electronics and data interchange.

The Telebyte Packetape tape drive subsystem is available in a desktop enclosure whose dimensions are nine inches wide × five inches high × 15 inches deep. The unit operates from 115/230 V, 50-60 Hz and consumes 70 W. The PC Packetape has a retail price of $2990. The electronics package is also available as a subassembly for inclusion in OEM applications. Delivery of Packetape is from stock to 2 weeks. Telebyte Technology Inc. is located at 270 E. Pulaski Road, Greenlawn, NY 11740; (800) 835-3298.

Reader Service Number 20

Software integrates data acquisition, analysis

Macmillan Software Company has announced a software package capable of converting an IBM PC, AT, XT, or compatible into a scientific workstation with fully integrated data acquisition, analysis, and graphics capabilities.

Asyst scientific software utilizes the PC's 8087 coprocessor to offer 80-bit internal precision for computational accuracy and a 1024-point fast Fourier transform that is said to take less than three seconds.

The three-module Asyst package consists of:

- System/Graphics/Statistics, which establishes the system environment, stores data, and provides graphics and basic statistics and math functions.
- Analysis, which reduces, manipulates, and analyzes data and provides advanced mathematical capabilities.
- Acquisition, which allows interface with scientific instruments to capture data directly with standard commands and minimal keyboarding.

Preprogrammed commands can be combined and modified to extend the capability of the system for custom applications. Conceptual language and an on-line "help" system are also built in.

Macmillan Software Company is located at 866 Third Ave., New York, NY 10022; (800) 348-0033 (in New York, (212) 702-3241).

Reader Service Number 21

For more information, circle the appropriate Reader Service Number on the Reader Service Card at the back of the magazine.

Expert systems tool kit available for UNIX PCs

A software tool kit for building and operating industrial-scale expert systems on the IBM PC-AT and PC-XT is now available from Radian Corporation.

Called RuleMaster, the new software package permits the creation of expert systems for advisory, diagnosis, prediction, or control applications. RuleMaster does not require Lisp or Prolog language skills or machines.

RuleMaster applications cover engineering, technical, and industrial areas, including fault diagnosis, on-line operations advice, interactive maintenance manuals, weather prediction, and chemical analysis advice, the company claims.

According to Radian, RuleMaster offers an approach to expert systems development which differs radically from that of systems based on conventional AI languages such as Lisp. Unlike Lisp, RuleMaster provides to knowledge engineers a pair of application development tools: RuleMaker, a facility for inducing rules from examples; and Radial, a high-level language for expressing rules. Radial is similar to structured algorithmic programming languages such as Ada and Pascal, the company claims.

The RuleMaker facility speeds application development by automatically generating rules and Radial code from a set of declarative examples supplied by an expert. The knowledge engineer gather examples from the expert and enters them into an "example table" in any order. RuleMaker induces procedural rules from the examples, and generates modules that express those rules in the Radial language. Conflicts in the resulting logic that stem from improperly constructed example tables are discovered and reported by RuleMaster to the user for correction. Additionally, omission of required logical entries in an example table are reported to the user for modification.

Knowledge engineers working with RuleMaster can write code directly in the Radial language. Manually written Radial code can be inserted into code generated by RuleMaker from an example table. RuleMaster can also access external routines written in various languages such as Fortran and Pascal.

The Radial language implements two fundamental reasoning mechanisms: backward and forward chaining. Most practical applications require a combination of backward and forward chaining for different parts of the solution. With RuleMaster, a knowledge engineer can combine both chaining methods in a single expert system.

Backward chaining is commonly employed for diagnostic applications. This reasoning mechanism is useful for selecting the best solution to a problem from a number of possibilities, such as the most probable cause of equipment failure from many possible causes.

Forward chaining is a reasoning mechanism useful in planning and monitoring expert systems. Applications include process control, robotic control, and simulation.

RuleMaster is available for use on the IBM PC-AT and PC-XT running under the Xenix or DOS 3.0 operating system. The price for the PC-AT version is $15,000; for the PC-XT version, $5000. Quantity discounts are available. The purchase price covers a four-day training session held at Radian each month.

Written in C, RuleMaster can potentially be run on any computer with a C compiler. The UNIX version of RuleMaster currently runs on DEC VAX systems, and has been implemented on several other UNIX-based systems from Gould, Perkin-Elmer, and Sun Microsystems. The UNIX version of RuleMaster is priced at $25,000.

Radian Corporation is located at 8501 Mo-Pac Blvd., PO Box 9948, Austin, TX 78766; (512) 454-4797.

Reader Service Number 22

Free Catalog!

Your 80-page guide to computer supplies and accessories–including complete new product descriptions.



- Packed with over 1600 products for microcomputers, minicomputers, and word processors – many available nowhere else.
- Big special section devoted to new supplies and accessories.
- Comprehensive product descriptions – including more than 475 full-color photos – clearly explain features and benefits.
- Easy-to-use cross reference guides to magnetic media, ribbons, and more–along with the industry's most complete cable guide.
- Helpful suggestions and tips, ranging from flexible disk care to proper ribbon selection to useful application ideas.

Phone toll-free 1-800-547-5444
In California, call 1-800-547-5447.

inmac™

Phone toll-free 1-800-547-5444 or send coupon today.*

Inmac Catalog Dept.
2465 Augustine Drive
Santa Clara, CA 95054

Please rush my free copy of the Inmac Catalog. I understand there is no obligation whatsoever.

NAME
COMPANY
ADDRESS
CITY
STATE ZIP PHONE

7 5 1 1 0 2

*In California, call 1-800-547-5447.

Reader Service Number 2

Reader Interest Survey

Indicate your interest in this department by circling the appropriate number on the Reader Interest Card.

High 188 Medium 189 Low 190

Product Summary

Editor: Victor P. Nelson

For more information, circle the appropriate Reader Service Number on the Reader Service Card at the back of the magazine.

MANUFACTURER	MODEL	COMMENTS	RS NO.
Communications			
Anchor Automation 6913 Valjean Avenue Van Nuys, CA 91406 (818) 997-7758	Signalman Computer Mailbox	Modem message center has the capability of storing messages for off-line micros and PCs, with up to 64K-byte memory. Features three levels of password security, remote message access. Priced at $299.	31
Code-A-Phone	Tel-A-Modem	Intelligent modem device integrated into two-line telephone offers voice/data communication capability. Available retail at $595.	32
Method Systems, Inc. 3511 Lost Nation Road Willoughby, OH 44094 (216) 942-2100	PCT-100 programmable translator system	User-programmable in-line RS-232 protocol and data translator may be used as a custom-configurable data security unit. With translator in-line, access can be limited to devices with similarly programmed code convertor. Priced at $495. Technical manual available for $25.	33
Micro Power Systems 3100 Alfred Street Santa Clara, CA 95054 (408) 727-5350	MP212A modem	Two-chip CMOS product uses <300 mW power and ±5V sources, provides Bell 212A, 103, and 113 and CCITT V.22 A,B and V.21 modem capability, plus dual-tone multifrequency generation, call-progress tone-detection, internal Universal Asynchronous Receiver and Transmitter (UART), and timing mechanisms for automatic modem handshaking. Available for $65-$75, depending upon interface.	34
XYPLEX, Inc. 100 Domino Drive Concord, MA 01742 (617) 371-1400	Step 1 communications system	Designed to link a DEC VAX to PCs and dumb terminals, system uses eight-port rack-mounted cluster controllers compatible with any RS-232 asynchronous device. $8100 for 16 connections, expandable to 64 connections at $2200 additional per eight ports.	35
Chips/Components			
Dallas Semiconductor 4350 Beltwood Parkway Dallas, TX 75234 (214) 450-0400	DS1217 NVRAM cartridge series	User-insertable nonvolatile CMOS RAM, packaged in 16-to-256K cartridge formats, is said to offer data retention for up to ten years by means of redundant lithium energy-source back-up. Power-switching circuitry initiates at <3.0 Vcc. Ribbon cable adaptor mates with JEDEC 28-pin bytewide socket. 16K model priced at $19.50 per unit for 100 units.	36
Peripherals			
Bering Industries, Inc. 1400 Fulton Place Fremont, CA 94539 (415) 651-3300	TopSecret-1 and -2 disk subsystems	System designed with the Hewlett-Packard HP-IB (IEEE-488) interface features a 10M-byte removable Winchester cartridge disk and optional 3.5-inch floppy disk drive. Applications include data security and disk backup. Priced at $3890; $4190 with floppy drive.	37
MiniScribe Corporation 1861 Lefthand Circle Longmont, CO 80501 (303) 651-6000	Models 8212 and 8425 micro disk drives	Designed for portable PCs, 3.5-inch Winchester drives provide 12.8M or 25.6M bytes of unformatted capacity.	38

Manufacturer	Model	Comments	Rs No.
Software			
AGS Computers, Inc. Advanced Products Division 1139 Spruce Drive Mountainside, NJ 07092 (201) 654-4321	Smart/C development environment	Software tool is said to be an integrated, precompilation development environment for the C language that incorporates AI principles. Consists of two modules: the Environment, a syntax-directed editor and interpreter; and the Migrator, a maintenance tool for programs not created with the Environment. Runs on IBM PC, AT&T 3B, and VAX 11/780 (among others) under MS-DOS, UNIX, XENIX, Ultrix. $500-$10,000, depending upon system.	39
Radian Corporation 8501 Mo-Pac Blvd. PO Box 9948 Austin, TX 78766-9948 (512) 454-4797	RuleMaster expert systems tool kit	Package for UNIX-based systems, written in C, permits expert systems developer to implement forward- and backward-chaining reasoning mechanisms. Consists of two development tools: RuleMaker, a facility for inducing rules from examples; and Radial, a high-level language for expressing rules. Developer working from RuleMaker writes rules directly in Radial, or can access external routines written in Fortran and Pascal. $5000-$25,000, depending upon system.	40
Systems			
Grid Systems Corp. 2535 Garcia Avenue Mountain View, CA 94043 (415) 961-4800	GRiDCase portable PC	This IBM-PC compatible is said to offer up to 512K RAM, 512K ROM, internal 1200/300 baud modem, 8087 math coprocessor, 640 × 200 bit-mapped flat-panel display, and local- and remote-area networking through GRiD Server, the company's networking system. Priced beginning at $2975, depending upon configuration.	41
Inner Access Corp. 517K Marine View Belmont, CA 94002 (415) 591-8295	MultiUser-16 multiuser/ multitasking system	Using the Motorola 68000 or 68010, system features 0.5M bytes of memory (configurable up to 16M bytes), 40M-byte hard disk (expandable to 1G bytes), eight-slot motherboard (optional 20 slots), 8-MHz operation (10-MHz optional), and optional 45M-byte tape backup. Runs under Mirage operating system. Price begins at $8995.	42
Micro Craft Corp. 4747 Irving Blvd., Suite 214 Dallas, TX 75247 (214) 630-2562	Dimension 68000 UNIX system	Using the Motorola 68000 under Uniplus+, the Unisoft implementation of UNIX System V, system priced at $15,995 includes 1M bytes of RAM with two 400K-byte 5¼ inch drives, 50M-byte hard disk, four serial ports, and four terminals with 25-foot cables. Can also be configured as a single-user workstation for process-intensive applications, priced from $3995-$7995.	43
Opus Systems Suite 120 960 San Antonio Road Los Altos, CA 94022 (415) 941-7201	Opus Personal Mainframe	Designed to integrate DOS and UNIX environments in IBM PC, AT&T 6300, Compaq Plus or Deskpro, and TI Professional Computer, system consists of Opus5, company's port of UNIX System V, and Opus16, 32-bit coprocessor with up to 2M bytes of on-board memory. Based on NS 32000 chip set.	44

Calendar

Conferences sponsored or cosponsored by the IEEE Computer Society are indicated by the society's logo. For listing in Calendar, submit information **eight weeks before cover date.** Send to Calendar, *IEEE Micro,* 10662 Los Vaqueros Circle, Los Alamitos, CA 90720-2578.

July 1985

World Computer Graphics 85 (WCGA, NCGA), July 10-12, New York, New York. Contact World Computer Graphics Association, 2033 M St., NW, Suite 399, Washington, DC 20036; (202) 775-9556; telex WORLD GRAPHICS 248301 RCAW UR, or National Computer Graphics Association, 8401 Arlington Blvd., Suite 601, Fairfax, VA 22031; (703) 698-9600.

Fourth Annual National Conference of the Association for Women in Computing, July 13-14, Chicago, Illinois. Contact Joan Wallbaum, 407 Hillmore Dr., Silver Spring, MD 20901.

NATO Advanced Study Institute on Solid-State Devices in Communications (IEEE), July 14-27, Erice, Sicily, Italy. Contact Martin V. Schneider, AT&T Bell Laboratories, HOH L-261, PO Box 400, Holmdel, NJ 07733; (201) 949-2503.

NCC 85, National Computer Conference (ACM, AFIPS, DPMA, SCS), July 15-18, Chicago, Illinois. Contact AFIPS, 1899 Preston White Dr., Reston, VA 22091; (703) 620-8900.

Siggraph 85, 12th Annual Conference on Computer Graphics and Interactive Techniques (ACM), July 22-26, San Francisco, California. Contact Siggraph 85 Conference Services Office, 111 E. Wacker Dr., Suite 600, Chicago, IL 60601; (312) 644-6610.

WCCE 85, World Conference on Computers in Education (ACM, IFIP, AFIPS), July 29-August 2, Norfolk, Virginia. Contact Richard L. Dobson, Jr., AFIPS, 1899 Preston White Dr., Reston, VA 22091; (703) 620-8935; TWX 710-833-9037.

CADCE 85, Third IFAC/IFIP International Symposium on Computer-Aided Design in Control and Engineering Systems (IEEE), July 31-August 2, Copenhagen, Denmark. Contact Danish Automation Society, Bldg. 229, The Technical University of Denmark, DK-2800 Lyngby, Denmark.

August 1985

ASME 85, International Computers in Engineering Conference and Exhibition, August 4-8, Boston, Massachusetts. Contact Steve Rhode, Power Systems Research Dept., General Motors Research Laboratories, Warren, MI 48090-9055; (313) 575-3004 or (313) 492-6635.

Fourth ACM Sigact-Sigops Symposium on Principles of Distributed Computing, August 5-7, Minaki, Ontario, Canada. Contact Michael A. Malcolm, Faculty of Mathematics, Dept. of Computer Science, University of Waterloo, Ontario N2L 3G1, Canada; (519) 885-1211.

IJCAI 85, International Joint Conference on Artificial Intelligence, August 18-24, Los Angeles, California. Contact American Association for Artificial Intelligence, 445 Burgess Dr., Menlo Park, CA 94025; (415) 321-1118.

14th Annual International Conference on Parallel Processing (ACM), August 20-23, St. Charles, Illinois. Contact T. Feng, Pennsylvania State University, EE East Bldg., University Park, PA 16802.

VLSI 85, International Conference on Very Large Scale Integration (IFIP), August 26-28, Tokyo, Japan. Contact VLSI 85, c/o Business Center for Academic Societies-Japan, 4-16 Yayoi, 2 Chome, Bunkyo-ku, Tokyo 113, Japan; phone (03) 815-1903.

ICED 85, International Conference on Engineering Design, August 26-29, Hamburg, West Germany. Contact VDI-Gesellschaft Entwicklung Konstruktion Vertrieb, Postfach 11 39, 4000 Dusseldorf I, West Germany.

September 1985

Euromicro, 11th Symposium on Microprocessing and Microprogramming, September 3-6, Brussels, Belgium. Contact Euromicro, Attn. Chiquita Snippe-Marlisa, p/a TH Twente, Dept. INF, Room A306, PO Box 217, 7500 AE Enschede, The Netherlands; phone +(31) (53)-338799; telex 44200 thes.

Office Automation Society Workshop and Third Annual Conference, September 3-6, Bloomington, Minnesota. Contact Office Automation Society International, 2108 C Gallows Rd., Vienna, VA 22180; (703) 790-0490.

IFIP Working Group 8.1 Working Conference on Environments to Support Information System Design Methodologies, September 4-6, Bretton Woods, New Hampshire. Contact Peter C. Lockemann, Institut für Informatik, Universität Karlsruhe, Zirkel 2, 7500 Karlsruhe 1, West Germany; phone 49 + (0721) 608-3968.

IPRC 85, International Personal Robot Congress and Exposition, September 6-8, San Francisco, California. Contact National Personal Robot Association, PO Box 1366, Dearborn, MI 48121; (313) 271-7800.

Compint 85, First International Conference on Computer-Aided Technologies (ACM), September 10-12 (tutorials to be offered September 9), Montreal, Canada. Contact Stephen G. Leahey, PO Box 577, Desjardins Postal Station, Montreal, PQ H5B 1B7, Canada; (514) 870-8842.

International Conference on Picture Archiving and Communication Systems (PACS IV) for Medical Applications (SPIE), September 10-12, Tokyo, Japan. Contact SPIE, PO Box 10, Bellingham, WA 98227-0010; (206) 676-3290.

Ninth Data Communications Symposium (ACM), September 10-13 (tutorials to be offered September 9), Whistler Mountain, British Columbia, Canada. Contact W. P. Lidinsky, AT&T Bell Laboratories, 6B309, Naperville-Wheaton Rd., Naperville, IL 60566; (312) 979-6246, or Ninth Data Communications Symposium, PO Box 639, Silver Spring, MD 20901; (301) 589-8142; TWX 7108250437 IEEECOMPSO.

NATO Advanced Study Institute on Intelligent Decision Aids in Process Environments, September 16-27, Pisa, Italy. Contact David Woods, Westinghouse R&D Center, Pittsburgh, PA 15235.

IMA Conference on Mathematics in Signal Processing (IEE), September 17-19, Bath, England. Contact Deputy Secretary, Institute of Mathematics and Its Applications, Maitland House, Warrior Square, Southend-on-Sea, Essex SS1 2JY, UK.

UNIX Expo, September 18-20, New York. Contact National Expositions Co., 14 West 40th Street, New York, NY 10018; (212) 391-9111.

International Seminar on Computer Networking and Performance Evaluation (INRIA, IFIP, et al.), September 18-20, Tokyo,

Japan. Contact Yutaka Takahashi, Dept. of Applied Mathematics and Physics, Kyoto University, Kyoto 606, Japan.

Informatica 85, 18th National Informatics Congress and Fifth International Computer Exhibition (IEEE), September 23-29, Sao Paulo, Brazil. Contact IEEE Sao Paulo Section, Ave. Sao Gabriel 555, Sala 407, 01435 Sao Paulo, SP, Brazil.

October 1985

Office Automation 85 (ACM), October 2-4, Erlangen, West Germany. Contact H. Wedekind, IMMD VI, Martensstr. 3, D-8520 Erlangen, West Germany.

23rd Annual Allerton Conference on Communication, Control and Computing, October 2-4, Monticello, Illinois. Contact Bruce Hajek or David C. Munson, Jr., Coordinated Science Laboratory, University of Illinois at Urbana-Champaign, 1101 W. Springfield Ave., Urbana, IL 61801.

ISDN 85, Integrated Services Digital Networks (IEEE), October 6-10, Destin, Florida. Contact Russell G. DeWitt, Contel Service Corp., Network Design, 245 Perimeter Center Pkwy., Atlanta, GA 30346; (404) 391-1895.

10th Conference on Local Computer Networks, October 7-9, Minneapolis, Minnesota. Contact Dan E. Gahlon, Interactive Systems/#3M, 225-4S-06 3M Center, St. Paul, MN 55144.

Intelec 85, International Telecommunications Energy Conference (IEEE), October 14-17, Munich, West Germany. Contact Gunther Vau, Siemens A.G., Postfach 3240, Erlangen 2, West Germany.

ASIS 85, October 20-25, Las Vegas, Nevada. Contact American Society for Information Science, 1010 16th St., NW, Washington, DC 20036.

Intersociety Conference on Artificial Intelligence Applications (IEEE), October 21-23, Washington DC. Contact B. G. Silverman, Institute for Artificial Intelligence, Library, Room 636, George Washington University, Washington DC 20052.

Infomatics 85, October 21-24, Amsterdam, The Netherlands. Contact Program Committee, Infomatics 85, PO Box 34404, Bethesda, MD 20817.

First Pacific Computer Communication Symposium (ACM et al.), October 22-24 (tutorials to be offered October 21), Seoul, Republic of Korea. Contact K. H. Kim, University of South Florida, Tampa, FL 33620; (813) 974-4184.

Northcon 85 (IEEE), October 22-24, Portland, Oregon. Contact Electronic Conventions Management, 8110 Airport Blvd., Los Angeles, CA 90045; (213) 772-2965; telex 181350.

Protext II, Second International Conference on Text Processing Systems, October 23-25, Dublin, Ireland. Contact Protext 2 Organizing Committee, c/o Boole Press, Ltd., PO Box 5, 51 Sandycove Rd., Dun Laoghaire, Co. Dublin, Ireland; phone (+353-1) 808025; telex 30547 SHCN El (Ref. BOOLE).

Fourth IEEE Acoustics, Speech, and Signal Processing Society Workshop on Multidimensional Signal Processing, October 28-30, Leesburg, Virginia. Contact Barbara B. Hope, Center for Automation Research, University of Maryland, College Park, MD 20742; (301) 454-4526.

Cadcon Central, Conference on Computer-Aided Engineering and Computer-Aided Design, October 29-31, Chicago, Illinois. Contact Morgan Grampian Expositions Group, 1050 Commonwealth Ave., Boston, MA 02215.

November 1985

Workstation 85, First International Conference and Exhibition on Computer Workstations, November 11-14, San Jose, California. Contact Edward Miller, Software Research Associates, 580 Market St., Suite 350, San Francisco, CA 94104; (415) 957-1441, or Workstation 85, PO Box 639, Silver Spring, MD 20901; (301) 589-8142; TWX 7108250437 IEEECOMPSO.

ICCAD-85, 1985 IEEE International Conference on Computer-Aided Design, November 18-21, Santa Clara, California. Contact Paul Weil, Hughes Aircraft Co. (MS 270/055), 8433 Fallbrook Ave., Canoga Park, CA 91034; (818) 702-1791.

Tutorial Week Washington, DC, November 18-22, Arlington, Virginia. Contact Tutorial Week-Washington, IEEE Computer Society, PO Box 639, Silver Spring, MD 20901; (301) 589-8142; TWX 7108250437 IEEECOMPSO.

Tutorial Week West, November 18-22, San Francisco, California. Contact IEEE Computer Society, PO Box 639, Silver Spring, MD 20901; (301) 589-8142; TWX 7108250437 IEEECOMPSO.

Wescon 85, Western Electronics Show and Convention (IEEE), November 19-22, San Francisco, California. Contact Dale Litherland, Electronic Conventions, Inc., 8110 Airport Blvd., Los Angeles, CA 90045; (213) 772-2965.

December 1985

Globecom 85, IEEE Global Telecommunications Conference, December 2-5, New Orleans, Louisiana. Contact D. P. Dodd, Bell South Services, PO Box C360, fourth floor, Birmingham, AL 35283; (205) 321-3723.

Micro 18, Microprogramming Workshop, December 2-6, Pacific Grove, California. Contact Micro 18, PO Box 639; Silver Spring, MD 20901; (301) 589-8142; TWX 7108250437 IEEECOMPSO.

DEXPO West 85, December 11-13, Anaheim, California. Contact Project Manager, Expoconsul International, 55 Princeton-Hightstown Road, Princeton Junction, NJ 08550; (609) 799-1661.

Second Conference on Artificial Intelligence Applications: The Engineering of Knowledge-Based Systems, December 11-13, Miami Beach, Florida. Contact Artificial Intelligence Conference, PO Box 639, Silver Spring, MD 20901.

Sixth International Conference on Information Systems (ACM), December 16-18, Indianapolis, Indiana. Contact Jeffrey A. Hoffer, Graduate School of Business, Indiana University, 10th and Fee Lane, Bloomington, IN 47405; (812) 335-8449.

First International Workshop on VLSI Design, December 26-28, Madras, India. Contact H. N. Mahabala, Dept. of Computer Science and Engineering, Indian Institute of Technology, Madras 600036, India, or Vishwani Agrawal, AT&T Bell Laboratories, Murray Hill, NJ 07974; (201) 582-4349.

MicroCourses

Editor: James J. Farrell III

IEEE Micro accepts announcements of short-course listings pertaining to microprocessor and system design. Announcements should designate course titles, locations, dates, costs, and contact address and telephone number. Information should be received at the address below at least two months **before** cover date, and should be current at least two months **after** cover date. Send announcements to MicroCourses, Dept. M-SC, *IEEE Micro*, 10662 Los Vaqueros Circle, Los Alamitos, CA 90720-2578.

Networking the IBM PC; Data Communications Systems; Office Automation; Selecting a PBX System; Unix/Xenix; Network Communications Protocols; SNA; LANs; courses held through August in various US locations; $695-745. **Contact: Center for Advanced Professional Education, 1820 East Garry Street, Suite 110, Santa Ana, CA 92705; (714) 261-0240.**

Computer Graphics, July 15-19; *Operating Systems,* August 5-9; *Advanced Project Management and Leadership,* August 12-16; *Fast Turnaround LSI Design Methodologies,* August 14-16; *Computer-Aided Geometric Design and Modeling,* August 26-30; *Telecommunication and Computer Communication Networks,* August 26-29; $745-895; discount for five or more employees. **Contact: Institute in Computer Science, University of California Extension, Santa Cruz, CA 95064; (408) 429-4534.**

Introduction to Unix; Text Processing in Unix; Shell Programming; Using C with Ultrix-11; Using C with Ultrix-32; self-paced instruction programs; on-site lectures/labs also available. **Contact: Digital Equipment Corporation, Educational Services, 12 Crosby Drive BUO/E55-54, Bedford, MA 01730; (800) 332-5656, ext. 005.**

Microelectronic Circuit Packaging; Microprocessor Hardware, Software, and Interfacing; Pascal, Ada, and C in Microelectronic Design; Advanced Microprocessor System Design; courses held through July in various Canadian locations; $115-$475; in-house training available. **Contact: Ontario Centre for Microelectronics, Suite 400, 1150 Morrison Drive, Ottawa, Ontario K2H 9B8; (613) 596-6690.**

Fiber Optic Communications, June 26-28; *Digital Image Processing and Applications,* July 15-18; *Methods for Management of Productivity and Quality,* July 30-August 2, August 27-30, December 3-6, December 10-13; *Digital Communications and Computer Network Systems,* August 5-8; *Integrating Word Processing, Data Processing, and Telecommunications to Increase Productivity,* August 5-9; *Life Testing and Burn-In Techniques,* August 12-15; *Video Disc Technology,* August 19-22; *Modern Electronic Interconnection and Packaging Systems,* September 26-27; *Decision Support Systems,* October 7-9; *Magnetic Recording Materials,* October 8-10; *Digital Magnetic Recording,* October 11-12; *Systems Analysis Techniques,* November 14-15; *Electronic Reliability Screening,* November 18-21; courses held in Washington, DC, and San Diego, CA; $650-835. **Contact: Continuing Engineering Education, George Washington University, Washington, DC 20052; (800) 424-9773.**

C Programming Workshop; Unix for Managers; Hands-On Unix for Programmers; courses held through July; $100-$750. **Contact: SSC, PO Box 7, Northgate Station, Seattle, WA 98125; (206) 367-8649.**

Pick System Fundamentals; Pick Basic; Advanced Pick Programming; courses held through August; $895. **Contact: Pick Systems Education Department, 1691 Browning, Irvine, CA 92714; (714) 261-7425.**

Robotics: Concepts, Theory, and Applications, June 24-28; *Principles of Microcomputers and Microprocessors,* July 15-19; *Written Communications for Engineers, Technical Writers, and Managers,* July 22-26; *Computer Vision and Image Processing,* July 29-August 2; *Contemporary Data Communication Networks,* August 5-9; $350-$975. **Contact: Engineering Summer Conferences, University of Michigan, 800 Chrysler Center, North Campus, Ann Arbor, MI 48109; (313) 764-8490.**

Data Communications System Components; Network Design, Operations, and Management; Network Protocols and Standards; SNA; LANs; Digital PBXs; X.25 and Packet Switching Networks; courses held through July in various US locations; $750-1350. **Contact: Systems Technology Forum, 9000 Fern Park Drive, Burke, VA 22015; (800) 336-7409.**

Micros for Managers: Software, eight videotape lectures available for lease or purchase. **Contact: Engineering Renewal and Growth, Colorado State University, Fort Collins, CO 80523; (800) 525-4950.**

Personal Computer and STD Computer Interfacing for Scientific Instrument Automation, August 22-24, Washington, D.C.; September 19-21, Greensboro, N.C.; $450. **Contact: CEC, Virginia Polytechnic Institute, Blacksburg, VA 24061; (703) 961-4848.**

Database Management and Fourth Generation Languages for Personal Computers; Data Communications: Network Design, Integration, and Applications; Data Communications and Networking for the IBM PC and Other Personal Computers; Fourth Generation Data Management Software; courses held through August in various US locations; $695-895. **Contact: Software Institute of America, 8 Windsor Street, Andover, MA 01810; (617) 470-3880.**

Microprocessor Fundamentals; Microprocessor Troubleshooting; Microprocessor Real-Time Interfacing and Control Systems; 16-Bit Microprocessors; Software Engineering for Micro and Minicomputer Systems; Software Series; Networks and Datacomm; Digital Image Processing; courses held through July in various US locations; videotape training series, self-study courses, and on-site courses also available. **Contact: Integrated Computer Systems, 6305 Arizona Place, Los Angeles, CA 90045; (800) 421-8166 outside CA; (800) 352-8251 inside CA.**

Reader Interest Survey

Indicate your interest in this department by circling the appropriate number on the Reader Interest Card.

High 191 Medium 192 Low 193

Access

Recent books and articles on microcomputing

Editor: Peter R. Rony/Department of Chemical Engineering/University of Delaware/Newark, DE 19716

Books

G. Phillips and D. J. Scellato, *Apple Macintosh Encyclopedia,* Methuen Inc., 1984, $19.95.

R. Moll and R. Folsom, *Macintosh Pascal,* Houghton Mifflin Co., 1985, $23.95.

K. Hudson, *Introducing CAL: A Practical Guide to Writing Computer-Assisted Learning Programs,* Methuen Inc., 1985, $19.95.

E. R. Tufte, *The Visual Display of Quantitative Information,* Graphics Press, 1983, $34.00. A superb book.

Articles

Business Computer Systems

Vol. 4, No. 3, Mar. 1985:

E. Horwitt, "Exploring Expert Systems," pp. 48-57.

T. Woods, "Computers Learn to Listen," pp. 80-83.

S. Austin, "Word Processing Programs: Bundles of Functions," pp. 85-100.

Vol. 4, No. 4, Apr. 1985:

E. Horwitt, "New Solutions to the Micro-to-Mainframe Puzzle," pp. 42-48.

A. Bernstein, "Innovating With Shop-Floor LANs," pp. 60-67.

D. Stewart, "Computer Conferences Come to Order," pp. 80-84.

Byte

Vol. 10, No. 3, Mar. 1985:

S. Ciarcia, "Build the Touch-Tone Interactive Message System," pp. 99-111.

P. Rice, "Arithmetic on Your PC," pp. 119-124.

R. S. Shuford, "Two Flat-Display Technologies," pp. 130-134.

F. N. Rounds, "Navigation: Putting the Microcomputer to Work at Sea," pp. 141-146.

D. L. Kahn, "A Unit-Conversion Algorithm," pp. 151-164.

Special section on bargain computing. Articles include "Build Your Dream Editor," "The Commodore 64 80-Column Terminal," "Public Domain Gems," "An XLISP Tutorial," and "Budget 3-D Graphics," pp. 171-236.

Byte Reviews: "The Altos 586 With the XENIX Development System," "The NEC APC III," "Atari 800XL," "The KoalaPad," and "FriendlyWriter and Friendly Speller," pp. 247-294.

Vol. 10, No. 4, Apr. 1985:

S. Ciarcia, "Build the Home Run Control System: Part 1—Introduction," pp. 102-110.

C. R. Wilson, "Coprocessing in Modula-2," pp. 113-117.

J. Hawley, "A Million-Point Graphics Tablet," pp. 120-121.

Special section on expert systems. Articles include "Communication With Alien Intelligence," "The Quest to Understand Thinking," "The Lisp Tutor," "Proust," "Architectures for AI," "The Lisp Revolution," "The Challenge of Open Systems," "Vision," "Learning in Parallel Networks," "Connections," "Reverse Engineering the Brain," "The Technology of Expert Systems," and "Inside an Expert System," pp. 124-330.

Byte Reviews: "The ITT Xtra" and "Insight—A Knowledge System," pp. 338-347.

Byte Japan: "The Fifth Generation in Japan," pp. 401-406.

Computer

Vol. 18, No. 2, Feb. 1985:

F. Maruyama and M. Fujita, "Hardware Verification," pp. 22-32.

N. Suzuki, "Concurrent Prolog As an Efficient VLSI Design Language," pp. 33-40.

M. S. Lam and J. Mostow, "A Transformational Model of VLSI Systolic Design," pp. 42-52.

S. M. German and K. J. Lieberherr, "Zeus: A Language for Expressing Algorithms in Hardware," pp. 55-65.

S. Dasgupta, "Hardware Description Languages in Microprogramming Systems," pp. 67-76.

R. Piloty and D. Borrione, "The Conlan Project: Concepts, Implementations, and Applications," pp. 81-92.

M. Shahdad et al., "VHSIC Hardware Description Language," pp. 94-103.

B. I. Witt, "Parallelism, Pipelines, and Partitions: Variations on Communicating Modules," pp. 105-112.

Vol. 18, No. 3, Mar. 1985:

G. D. Buzzard and T. N. Mudge, "Object-Based Computing and the Ada Programming Language," pp. 11-19.

E. L. Greenberg, R. E. Malcho, P. J. Stoll, and D. J. Theis, "Survey of Spacecraft Memory Technologies," pp. 29-39.

J. N. Fenner, J. A. Schmidt, H. A. Halabi, and D. P. Agrawal, "Masco: The Design of a Microprogrammed Processor," pp. 41-53.

W. Myers, "An Assessment of the Competitiveness of the United States Software Industry," pp. 81-92.

Datamation

Vol. 31, No. 5, Mar. 1, 1985:

I. Fuerst, "Broken Windows," pp. 46-52.

M. Pykkonen, "Handicapping LANs," pp. 96-102.

Vol. 31, No. 6, Mar. 15, 1985:

W. Schatz, "Keeping Pirates at Bay," pp. 57-60.

R. J. Crutchfield, "Terminal Traumas," pp. 66-68.

O. Serlin, "Fault-Tolerant Blues," pp. 82-88.

L. Froehlich, "Babbage Observed," pp. 119-124.

Vol. 31, No. 7, Apr. 1, 1985:

I. Fuerst, "When $10K Equals $400,000," pp. 76-80.

J. W. Verity, "Graphically Speaking With Dr. Edward R. Tufte," pp. 88-92.

J. A. Meads, "Friendly or Frivolous?" pp. 96-100.

Vol. 31, No. 8, Apr. 15, 1985:

J. Lamb, "OSI—As Far Off As Ever?" pp. 67-70.

Dr. Dobb's Journal

Vol. 10, No. 3, Mar. 1985:

Special issue on Prolog. Articles include:

J. Malpas, "Programming in Logic," pp. 36-41.

D. E. Cortesi, "Tour of Prolog," pp. 44-63.

D. Schlobohm, "Tax Advisor: A Prolog Program Analyzing Tax Issues," pp. 64-92.

Vol. 10, No. 5, May 1985:

Special section on graphics algorithms. Articles include:

T. Hogan, "Using Decision Variables in Graphics Primitives," pp. 40-48.

R. Rylander, "Solid Shape Drawing on the Commodore 64," pp. 50-82.

Other articles include:

G. A. Edgar, "A Compiler Written in Prolog," pp. 84-96.

J. Kronman, "Review of Disk Maker I," pp. 100-106.

Electronics Week

Vol. 58, No. 7, Feb. 18, 1985:

D. M. Weber, "Consumer Gamble," pp. 61-67.

J. S. Kontur and L. Letham, "Locking Up System Security," pp. 68-72.

Vol. 58, No. 8, Feb. 25, 1985:

H. Bierman, "SMDs Tax Board Testers," pp. 49-54.

Vol. 58, No. 9, Mar. 4, 1985:

T. Manuel, "Unix Update: Coming of Age," pp. 47-52.

Vol. 58, No. 10, Mar. 11, 1985:

J. Lyman, "Reliability Gets Promotion," pp. 61-67.

D. Morris and R. W. Peters, "CMOS Chips Gain EE Memory," pp. 69-72.

Vol. 58, No. 11, Mar. 18, 1985:

S. D. Personick, "Switches Take to Optics," pp. 55-58.

R. Rosenberg, "Quiet Print Invades Office," pp. 59-63.

Vol. 58, No. 12, Mar. 25, 1985:

L. Gould, "Computers Run the Factory," pp. 55-60.

Vol. 58, No. 13, Apr. 1, 1985:

B. C. Cole, "Wafer-Scale Faces Pessimism," pp. 49-53.

L. Gould, "Computers Get It All Together," pp. 55-59.

Vol. 58, No. 14, Apr. 8, 1985:

J. Lyman, "Components for SMA Arrive," pp. 49-53.

M. Roth, "Mass Storage Gets New Boost," pp. 55-59.

IEEE Communications Magazine

Vol. 23, No. 3, Mar. 1985:

N. Q. Duc and E. K. Chew, "ISDN Protocol Architecture," pp. 15-22.

B. M. Leiner, R. Cole, J. Postel, and D. Mills, "The Darpa Internet Protocol Suite," pp. 29-34.

M. A. Sirbu and L. E. Zwimpfer, "Standards Setting for Computer Communication: The Case of X.25," pp. 35-45.

H. Rudin, "An Informal Overview of Formal Protocol Specification," pp. 46-52.

IEEE Control Systems Magazine

Vol. 5, No. 1, Feb. 1985:

W. F. Powers, "Computer Tools for Modern Control Systems Design," pp. 14-17.

H. A. Sutherland and K. L. Sonin, "Control Engineer's Workbench—A Methodology for Microcomputer Implementation of Controls," pp. 22-26.

R. P. Paul, "The Early Stages of Robotics," pp. 27-31.

A. A. Desrochers, "Introduction to Robot Dynamics and Control," pp. 32-34.

IEEE Spectrum

Vol. 22, No. 4, Apr. 1985:

G. Kaplin and E. J. Lerner, "Realism in Synthetic Speech," pp. 32-37.

S. S. Seth and V. D. Agrawal, "Cutting Chip-Testing Costs," pp. 38-45.

G. Zorpette, "Computers That Are 'Never' Down," pp. 46-54.

Vol. 22, No. 5, May 1985:

Best Bits—Applications of Microprocessors: "A Micro and More," "Need a Lift?" and "Preprocessor for Faster Graphics," p. 26.

Special issue entitled "At Home With High Technology." Articles include "The Superstructure: Designing for High-Tech," "Heating and Cooling," "Electronic Watchdogs," "C^3I for the Home Owner," "Lighting Sets the Scene," "Robots in the Home: Promises, Promises," "Portia's Perfect Pad: Superhigh Tech," "Impact 2000: A Place in the Sun," "Eaglecrest: A Commuter's Dream," "All the Comforts of Home," "Audio in the Living Room," "Epicurean High-Tech," "More Than Fun and Games," "Not Just for Sleeping," "Monitoring the Nursery," "Making Waves in the Bath," "The High-Tech Hobbyhorse," "Body Building Goes High-Tech," "The Way We Wash," "Satire: When High-Tech Goes Awry," and "To Probe Further," pp. 34-112.

IEEE Transactions on Industrial Electronics

Vol. IE-32, No. 2, Mar. 1985:

P. Papasratorn and P. Preapinmongkolkarn, "A Small-Scale Distributed Microprocessor System Using Shared Memory Technique," pp. 97-102.

P. K. Chande and P. C. Sharma, "Microprocessor-Based Flow Measurement System," pp. 103-107.

J. E. Roehl, "A Microprocessor-Controlled Chemical Detection and Alarm System Based on Ion Mobility Spectrometry," pp. 108-113.

N. Chaudhuri, S. Ghosh, and A. M. Ghosh, "A Technique for Simultaneous Measurement With a Microcomputer," pp. 114-119.

R. E. Betz and R. J. Evans, "Microprocessor Control of a Cycloconverter," pp. 120-129.

"Interpolations, Differentiation, Data Smoothing, and Least Squares Fit to Data With Decreased Computational Overhead," pp. 135-141.

Macworld

Vol. 2, No. 4, Apr. 1985:

G. McComb, "Pictures to Pixels," pp. 68-79.

N. Lavroff, "Computer Chess Comes Alive," pp. 84-89.

Microcomputer Applications

Vol. 2, No. 3, 1983:

E. T. Perez, G. J. Martinez, and B. F. Womack, "Microprocessor-Based Automatic Speed Control of a DC Motor Faced With Torque Disturbances," pp. 72-75.

K. Srinivasan and A. I. Merchang, "Digital Control of an Electrohydraulic Position Servomechanism Using a Microcomputer," pp. 76-80.

J.-M. Farines and M. C. Filho, "Variable Speed PWM Inverter-Fed Induction Motor Drive Using a Microcomputer," pp. 81-85.

D. Delmas, D. Vliegen, C. Marche, and G. Leclerc, "Development of a Low-Cost Microcomputer-Based System for Tracer Dispersion Measurement," pp. 86-90.

Vol. 3, No. 2, 1984:

D. I. Moldovan and G. K. F. Lee, "On the Rapid Solution of Differential Equations by Microprocessors," pp. 37-44.

D. G. Harrington and T. C. Hsia, "Ultrasonic Ranging in a Computer-Controlled Mobile Robot," pp. 49-53.

Vol. 3, No. 3, 1984:

G. Ramachandran, P. A. Parker, and R. N. Scott, "Microprocessor-Based On-Line System for Monitoring Spinal Cord Integrity," pp. 55-59.

W. L. Cleghorn et al., "Computer-Aided Analysis and Graphics by Microcomputer—Manitoba Experience," pp. 60-65.

L. Halsted and M. L. Hull, "A Miniature Microcomputer-Based Ski Binding Controller," pp. 66-68.

D. D. Ardayfio, D. Lane, D. S. Fraction, and D. Fritsche, "Simulation of Robots in the Task Environment Using Leadthrough Programming," pp. 69-73.

D. L. Hudson, M. E. Cohen, and P. C. Deedwania, "Emerge, A Rule-Based Expert System Implemented on a Microcomputer," pp. 79-83.

Microprocessors and Microsystems

Vol. 8, No. 10, Dec. 1984:

R. W. King and N. Cope, "Real-Time Braille Interface for Videotex Interaction," pp. 528-534.

Vol. 9, No. 1, Jan./Feb. 1985:

S. M. Said and K. R. Dimond, "Powerful Trace Facility for the MC68000 Microprocessor System," pp. 3-7.

J. Bowen, "Software/Hardware Integration on Microprocessors," pp. 8-14.

P. K. Chakraborty and J. C. Biswas, "Definition of Unspecified Flag Bits of the Z80 Microprocessor," p. 15-16.

O. R. Omatayo, "Automatic Timing of User-System Interaction As an Evaluation Aid for Microsystems," pp. 17-20.

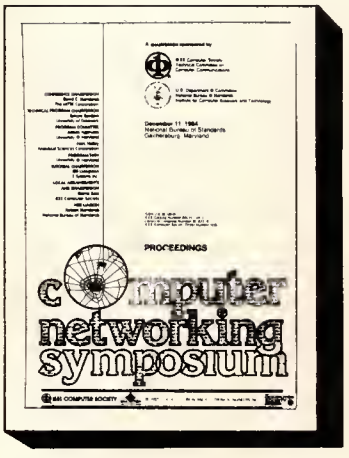
H. Al-Riahi, "Software-Controlled Memory Duplication," pp. 21-23.

Seventeen papers organized in six sessions with primary topics centering on local area network technology and protocol, performance modeling and analysis, evaluation and specification, distributed systems, and communications technology. 157 pp.

Computer Networking Symposium

December 11, 1984

Members—$16.00
Nonmembers—$32.00



Order #606
Handling Charges Extra
Order from IEEE Computer Society Order Dept.
PO Box 80452, Worldway Postal Center
Los Angeles, CA 90080 USA
(714) 821-8380

J. Bowen, "6800/6802 Microprocessor Emulation Adapter," pp. 24-26.

Mini-Micro Systems

Vol. 18, No. 3, Feb. 15, 1985:

Communications Digest issue. The five product categories that are covered are "Modems," "Multiplexers," "Local-Area Networks," "Networking Software," and "Voice/Data Computer/Terminals." Extensive product guides are provided.

Vol. 18, No. 4, Mar. 1985:

R. Dalrymple, "Tiny Single-Board PC May Spark Big Markets," pp. 33-35.

L. Valigra, "Some IBM Mimics Jump, Some Stall at PC AT Delays," pp. 34-43.

T. Moran and M. Tucker, "3Com Brings Network Servers to Small LANs," pp. 43-47.

L. Huffman, "Atari Strives for a Comeback With Macintosh Look-Alike," pp. 47-52.

C. Hintermeister, "Taiwanese Court Gives Light Fine to Apple Counterfeiters," pp. 65-68.

D. Danks, "Europeans Adopt Alternative OS as IBM Becomes Proprietary," p. 68.

D. Simpson, "Vendors Veer Toward VARs, Vertical Markets," pp. 73-80.

K. Jones, "Computer Standards: The Jury Is Still Out," pp. 85-88.

C. Warren, "Environmental Software Opens System Windows," pp. 115-131.

G. R. Talsky, "New Portable Computers Better Aimed at Market," pp. 135-143.

R. Dalrymple, "System Integrators Exploit PC Compatibility," pp. 147-155.

R. S. Harp, "'LAN in a Box' Provides Multiuser System," pp. 159-167.

J. J. Roloff, "Processor Board Handles Virtual Memory," pp. 173-178.

D. A. Dawson, "Foundation Module Simplifies Integration," pp. 183-188.

Vol. 18, No. 5, Apr. 1985:

L. Valigra, "ADAPSO Battles Micro Software Pirates," pp. 27-28.

J. P. Mello, "Intel's LAN Controller Chip Cuts Costs, May Ignite Sales," pp. 35-38.

J. Borrell and J. Donohue, "ACT Starts US Bid With Three Software Allies," pp. 41-43.

J. Isaak, "Supermicro Braves Harsh Environments," pp. 85-90.

M. Tucker, "Integrated Software Spurs Mini Market," pp. 95-104.

J. Victor, "Uninterruptible Power Supplies Win Converts Despite Obstacles," pp. 119-154.

Mundo Electrónico

No. 148, Mar. 1985:

J. M. Fuertes Armengol, "Sistemas Tolerantes a Fallos: Applicación a Máquinas de Transporte," pp. 55-60.

J. C. Costilla, J. M. Perez, and P. de la Cruz, "Bases de Datos: El Modelo Relacional," pp. 63-68.

J. J. Perez Lopez, "CI Semicustom: Una Alternativa al Acceso de Tecnología Punta," pp. 83-89.

PC Tech Journal

Vol. 3, No. 3, Mar. 1985:

W. Fastie, "Environmental Engineering: Which Operating System Will You Choose?" pp. 9-10.

D. Awalt, "Concurrent PC-DOS," pp. 45-54.

D. Daegwyler, "Professional Debugging," pp. 60-70.

A. Harian and J. Krantz, "Programming for the 3270-PC," p. 74-88.

D. Awalt, "Enhancement by JRAM-2," pp. 92-99.

E. Nisley, "Spinning Your Own VDISK," pp. 100-111.

W. H. Murray, "Digital Designs," pp. 112-121.

P. G. Aitken, "Plotting Data," pp. 123-129.

A. Chaturvedi, "Tree Structures: Part 2," pp. 131-158.

A. Hansen, "The Making of an XT," pp. 161-171.

M. S. Oppenheimer, "The Tort of Copy Protection," pp. 177-180.

Vol. 3, No. 4, Apr. 1985:

W. Fastie, "Microsoft Re-emerges," pp. 9-10.

Cybernetic Micro Systems, "Product of the Month: Sim8051," p. 25.

T. Mirecki, "Replacing RENAME," p. 41.

T. Forgeron, "We Interrupt This Program," p. 42.

R. M. Foard, "Uptown Printers: IBM's Quietwriter and Wheelprinter," pp. 49-55.

T. V. Hoffman, "Graphic Enhancement," pp. 58-77.

S. Armbrust, "Untangling Programs," pp. 81-93.

V. Mansfield, "Encryption Methods," pp. 96-114.

W. D. Schwaderer, "CRC Calculation," pp. 118-133.

A. A. Gleckler, "File-Search Help for PC-DOS," pp. 139-145.

T. McNamee, "Screen Composure," pp. 149-159.

T. W. Madron, "Searching With Soundex," pp. 163-168.

E. I. Lee, "Recursive Curves," pp. 171-174.

M. S. Oppenheimer, "Legal Brief: The Compatibility Risk," pp. 179-181.

PC World

Vol. 3, No. 4, Apr. 1985:

A. Hoenig, "Program Patchwork," pp. 52-57.

R. White, "Thinking Big on Silicon Prairie," pp. 87-92.

T. Datz, "A Brighter Star?" pp. 116-126.

D. McCune, "The Myth of the Virtual Device," pp. 178-187.

K. Yates, "Hi-Fi Floppy," pp. 190-196.

D. Jenkins, "Upwardly Mobile dBase," pp. 207-212.

E. Weiss, "Setting the Data Table," pp. 216-223.

Systems and Software

Vol. 4, No. 3, Mar. 1985:

S. Feldman, "Will DEC Stage Comeback With New Multiuser Micro?" pp. 40-42.

M. Chester, "Business Forms and the X.400 Protocol Help Drive E-Mail," pp. 61-63.

W. Rauch-Hindin, "Communication Standards: OSI Is Not a Paper Tiger," pp. 64-86.

B. Beach, A. Pender, and M. Szabados, "OSI Protocols Link PC-DOS, Xenix, and iRMX," pp. 89-95.

J. Eaglstun and R. Altizer, "Transport-Protocol Software, Simple and Efficient," pp. 99-101.

D. H. Czubek, "SNA Gets Peer Communications," pp. 103-105.

M. Cox, "Virtual-Terminal Service Resolves Incompatibilities," pp. 107-111.

D. Melin and B. Phillips, "Package Makes Different LANs Compatible," pp. 113-114.

M. Chester, "Fault-Tolerant Computers Mature," pp. 117-129.

I. Hutzel and C. Peters, "The Current State-of-the-Art in User Interfaces," pp. 131-140.

Vol. 4, No. 4, Apr. 1985:

J. Isaak, "/usr/Group, IEEE Committees Join to Develop Unix Standard," p. 44.

R. Bernhard, "More Losers Than Winners in the LAN Standards Game," pp. 65-66.

J. McLeod, "Who'll Win the 100-to-300-Mbyte Capacity Fight?" pp. 68-84.

System Spotlight: "Packages Span Ada's Growth," pp. 93-100.

G. Pelton, A. Gilberg, and M. Meredith, "Magix: A Real-Time Unix for Factory Automation," pp. 103-109.

J. H. Alexander and M. J. Freiling, "Smalltalk-80 Aids Troubleshooting System Development," pp. 111-118.

R. Patton, "An Application-Level Graphics Standard," pp. 121-126.

Product Update: "As Data-Comm Tasks Get Harder, Modems Get Smarter," pp. 129-133.

Addresses of publishers

Business Computer Systems
Cahners Publishing Co.
270 St. Paul St.
Denver, CO 80206
(303) 388-4511
(Publisher does not sell back issues or single copies of articles. Subscriptions are free to qualified readers.)

Byte
Byte Publications Inc.
70 Main St.
Peterborough, NH 03458
(603) 924-9281

Computer
IEEE Computer Society
10662 Los Vaqueros Cir.
Los Alamitos, CA 90720
(714) 821-8380

Datamation
Technical Publishing Co.
1301 S. Grove Ave.
Barrington, IL 60010
(312) 774-8115

Dr. Dobb's Journal
People's Computer Co.
2464 Embarcadero Way
Palo Alto, CA 94304
(415) 424-0600

Electronics Week
McGraw-Hill Publications Co.
1221 Ave. of the Americas
New York, NY 10020
(212) 512-2000

Graphics Press
Box 430
Cheshire, CT 06410
(203) 272-9187

Houghton Mifflin Co.
Wayside Rd.
Burlington, MA 02107
(617) 272-1500

IEEE Communications Magazine
IEEE Control Systems Magazine
IEEE Spectrum
IEEE Transactions (all)
IEEE Service Center
445 Hoes Lane
Piscataway, NJ 08854
(201) 981-0060, x133

Macworld
555 De Haro St.
San Francisco, CA 94107
(415) 861-3861

Methuen Inc.
733 Third Ave.
New York, NY 10017
(212) 922-3550

Microcomputer Applications
International Society for Mini and Microcomputers
Acta Press
Box 2481
Anaheim, CA 92804

Microprocessors and Microsystems
Business Press International
205 E. 42nd St., Suite 1705
New York, NY 10017
(212) 867-2080

Butterworth Scientific Ltd.
Journals Division
PO Box 63
Westbury House
Bury St.
Guildford, Surrey GU2 5BH UK

Mini-Micro Systems
Cahners Publishing Co.
270 St. Paul St.
Denver, CO 80206
(303) 388-4511

Mundo Electrónico
Boixareau Editores, SA
Grand Via de les Corts Catalanes 594
2 Barcelona 2 Spain

PC Tech Journal
Ziff-Davis Publishing Co.
1 Park Ave.
New York, NY 10016
(212) 725-7947

PC World
555 De Haro St.
San Franciso, CA 94107
(415) 861-3861

Systems and Software
Hayden Publishing Co., Inc.
50 Essex St.
Rochelle Park, NJ 07662
(201) 843-0550

Reader Interest Survey

Indicate your interest in this department by circling the appropriate number on the Reader Interest Card.

High 194 Medium 195 Low 196

Advertisers & Products

Advertisers

IEEE Computer Society Publications 101

IEEE Computer Society Membership Cover III

IMSL 78

INMAC, Inc. 93

For Display Advertising Information Contact:

Southern California and Mountain States: Richard C. Faust Company, 24050 Madison Street, Suite 100, Torrance, CA 90505; (213) 373-9604.
Northern California and Pacific Northwest: Don Farris Company, 161 W. 25th Ave., #102B San Mateo, CA 94403; (415) 349-2222.
Jack Vance, P.O. Box 3205, Saratoga, CA 95070, (408) 741-0354
East Coast: Hart Associates, Inc., P.O. Box 339, 42 Lake Blvd., Matawan, NJ 07747; (201) 583-8500.
New England: Arpin Associates, P.O. Box 227, Weston, MA 02193; (617) 899-5613.
George Watts, III, 4 Conifer Dr., Wilbraham, MA 01095; (413) 596-4747.
Midwest: Thomas Knorr, Knorr MicroMedia, Inc. 333 North Michigan Ave. Chicago, IL 60601; (312) 726-2633.
Southeast: Larry C. Shattles, 133 Laurel Oak Drive, Longwood, FL 32779; (305) 788-1950
Southwest: The House Company, 3817 Richmond Avenue, Suite 110, Houston, TX 77027; (713) 622-2868.

Advertising Director: Mike Koehler, *IEEE MICRO*, 10662 Los Vaqueros Circle, Los Alamitos, CA 90720, (714) 821-3240, 821-8380,

For production information, conference or classified advertising contact Sandra J. Arteaga, *IEEE MICRO*, 10662 Los Vaqueros Circle, Los Alamitos, CA 90720, (714) 821-1140.

Moving?

PLEASE NOTIFY US 4 WEEKS IN ADVANCE

MAIL TO:
IEEE Service Center
445 Hoes Lane
Piscataway, NJ 08854

Name (Please Print)

New Address

City State/Country Zip

ATTACH LABEL HERE

- This notice of address change will apply to all publications to which you subscribe.
- List new address above.
- If you have a question about your subscription, place label here and clip this form to your letter.

Products

	RS#	Page#
Boards		
Communications	7	86
Multibus Adaptor	17	91
Voice	10	88
Circuits/Components		
Cartridge Series	36	94
RAM	6	86
I/O-Related Equipment		
Drives	38	94
Modems	31,32,34	94
Optical Disc	18	91
Processor Convertor Chip	15	90
Other Products & Services		
Computer Supplies Catalog	2	93
Membership		Cover III
Publications		101
Software		
Data Acquisition & Analysis	21	92
Interactive Stat	9	87
Mathematics/Statistics	1	78
Product Summary	39-40	95
Systems		
Backup Tape	19	91
Cartridge Backup	20	92
Communications	35	94
Dual-Mode Graphics	14	90
Expert Systems	22	93
Hybrid	12	89
Micro Development	16	90
PhotoMail	13	89
Product Development	11	88
Product Summary	41-44	95
Programmable Translator	33	94
Subsystems	36	94
Supermicro	8	87

IEEE MICRO

For further information on advertised products, new products, or literature, fill out the **Reader Service Card** (top). Circle the number on the RS Card that corresponds to the number of the item for which you would like more information.

To indicate your interest in an article or department, fill out the **Reader Interest Card** (bottom). Circle the number on the RI Card that corresponds to the level of interest given in the Reader Interest Survey at the end of the article or department.

Please print or type your name and address.

READER SERVICE CARD

IEEE MICRO INFORMATION ABOUT PRODUCTS

Void after December 15, 1985

6/85

Name _____

Company _____

Address _____

City _____

State _____

Zip _____

Country _____

Title _____

Telephone number () _____

PRODUCTS PURCHASED OR SPECIFIED	FOR JOB	FOR HOBBY
Computers	☐	☐
Peripherals	☐	☐
Data communications equip.	☐	☐
Memories, components	☐	☐
Software and services	☐	☐
Publications	☐	☐
Other	☐	☐

☐ Please send me information on advertising in *IEEE Micro*.

Product announcements, and products for review, should be sent to Richard Landry, Managing Editor, *IEEE Micro*, 10662 Los Vaqueros Circle, Los Alamitos, CA 90720-2578.

Send more information on numbered items:

1	5	9	13	17	21	25	29	33	37	41	45	49	53	57	61	65	69	73	77	81	85	89	93	97
2	6	10	14	18	22	26	30	34	38	42	46	50	54	58	62	66	70	74	78	82	86	90	94	98
3	7	11	15	19	23	27	31	35	39	43	47	51	55	59	63	67	71	75	79	83	87	91	95	99
4	8	12	16	20	24	28	32	36	40	44	48	52	56	60	64	68	72	76	80	84	88	92	96	100

READER INTEREST CARD

IEEE MICRO EDITORIAL RESPONSE

6/85

Name _____

Company _____

Address _____

City _____

State _____

Zip _____

Country _____

Title _____

Telephone number () _____

Comments:

How do you like *IEEE Micro*'s new format? Is there anything you would like to see changed?

Continue comments on other side

Reader Interest Survey:

101	116	131	146	161	176	191
102	117	132	147	162	177	192
103	118	133	148	163	178	193
104	119	134	149	164	179	194
105	120	135	150	165	180	195
106	121	136	151	166	181	196
107	122	137	152	167	182	197
108	123	138	153	168	183	198
109	124	139	154	169	184	199
110	125	140	155	170	185	200
111	126	141	156	171	186	201
112	127	142	157	172	187	202
113	128	143	158	173	188	203
114	129	144	159	174	189	204
115	130	145	160	175	190	205

☐ Please send me the IEEE-CS *Publications Catalog*.

☐ Please send me an IEEE Fellow nomination form.

☐ Please send me an IEEE Computer Society membership application.

☐ Please send me an *IEEE Micro* author's guide.

☐ Please send me information about reviewing article submissions for *IEEE Micro*.

This PO box for reader service cards only.

PLACE STAMP HERE

IEEE MICRO

Reader Service Inquiries
Box 24168
Los Angeles, CA 90024
USA

Comments on articles and other editorial matter:

I liked _____

I disliked _____

I would like _____

For Reader Interest Survey, see other side

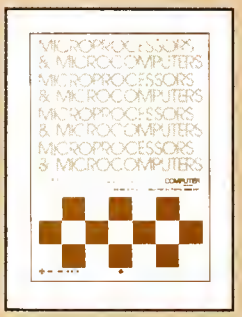
PLACE STAMP HERE

IEEE MICRO

10662 Los Vaqueros Circle
Los Alamitos, CA 90720-2578
USA

IEEE COMPUTER SOCIETY
Membership Application

Join now and receive one of these popular books absolutely free!





Microprocessors and Microcomputers—

All-new third edition — Selected papers from *Computer* and *IEEE Micro* (372 pp.)

Regular nonmember price: $20.00

Selected Reprints in Software—

Selected papers from *Computer* (282 pp.)

Regular nonmember price: $12.00

Offer does not apply to student member applicants.

Check here for your free book. Check one:
- ☐ *Microprocessors and Microcomputers*

or
- ☐ *Selected Reprints in Software*

Check box for additional information:
- ☐ Computer Society Membership Brochure
- ☐ Bulk air option (two-to-six-week delivery of publications) for members in Western Europe, Latin America, and the Far East
- ☐ Student Membership Application
- ☐ Publications Catalog

Schedule of Fees (tax deductible)

IMPORTANT: Pay the FULL-YEAR rate if your application is postmarked between September 1, 1984 and February 28, 1985. Pay the HALF-YEAR rate if your application is postmarked between March 1, 1985 and August 31, 1985. Membership expires December 31, 1985.

For applications received:	Half-Year 3/1/85–8/31/85	Full-Year 9/1/84–2/28/85
• **an IEEE member who wishes to join the Computer Society** IEEE membership no._____	☐ **$5.00**	☐ **$10.00**
• **a non-IEEE member who wishes to join both IEEE and the Computer Society*** Full IEEE members receive *Spectrum*, *The Institute*, and many additional benefits	☐ **$52.50**	☐ **$90.00**
• **a non-IEEE member who wishes to join the Computer Society only**	☐ **$17.00**	☐ **$34.00**

*Individuals who join both the IEEE and the Computer Society in Regions 7 (Canada) and 8 (Europe, Mid-East, & Africa) may deduct $6 from full-year rates/$3 from half-year rates; in Region 9 (Latin America), deduct $13/$6.50; in Region 10 (Asia & Pacific), deduct $12/$6.

ACM members who join both the IEEE and the Computer Society may deduct $5 from full-year rates/$2.50 from half-year rates.

Publication Options

If, in addition to your automatic subscription to *Computer* Magazine, you would like to receive:

	add:	
• *IEEE Computer Graphics & Applications* (3061)	☐ $11.00	☐ $22.00
• *IEEE Micro* (3071)	☐ $7.50	☐ $15.00
• *IEEE Design & Test of Computers* (3111)	☐ $6.00	☐ $12.00
• *IEEE Software* (3121)	☐ $7.00	☐ $14.00
• *Transactions on Computers* (1161)	☐ $7.50	☐ $15.00
• *Transactions on Software Engineering* (1171)	☐ $8.00	☐ $16.00
• *Transactions on Pattern Analysis & Machine Intelligence* (1351)	☐ $6.00	☐ $12.00
• *Journal on Lightwave Technology* (4301)	☐ $5.00	☐ $10.00
• *Journal of Robotics and Automation* (4401)	☐ $5.00	☐ $10.00
• *Journal of Solid State Circuits* (4101)	☐ $2.50	☐ $5.00
• *Proceedings of the IEEE* (5011) (available only to IEEE members)	☐ $7.50	☐ $15.00

Total amount remitted with this application $_____

☐ Check or money order enclosed. (Make payable to IEEE. Mail to IEEE Computer Society, 10662 Los Vaqueros Circle, Los Alamitos, CA 90720.)

☐ VISA ☐ MasterCard ☐ American Express

Charge Card Number

Exp. Date Mo. Yr.

I hereby make application for IEEE Computer Society membership and, if elected, will be governed by IEEE's and the Society's Constitutions, Bylaws, and Statements of Policies and Procedures.

Full signature _____ Date _____

MAILING ADDRESS

First Name _____ Middle initial(s) _____ Last name _____

Street address _____

City _____ State/Country _____ Zip _____

EDUCATION (highest level completed)

Course _____ Degree received _____ Date _____

Name of educational institution _____ Date of birth _____ ☐ Male ☐ Female

OCCUPATION

Title or position _____

Firm name _____

Firm address _____

City _____ State/Country _____ Zip _____

Affiliate society membership no. (if applicable) _____ Grade _____

REFERENCE (an IEEE member; if unknown, a managerial person who knows you professionally.)

Name (print in full) _____

Position _____

Street address _____

City _____ State/Country _____ Zip _____

MAIL TO: IEEE COMPUTER SOCIETY, 10662 LOS VAQUEROS CIRCLE, LOS ALAMITOS, CA 90720.

PMCR 685

UPCOMING ISSUES OF IEEE MICRO

AUGUST 1985

High-Level Languages/ Multiprocessing

- A Real-Time Fortran Executive
- Mapping High-Level Syntax and Structure into Assembler Language
- High-Level Language Programmable Controller

ALSO

- Mutual Exclusion/ Synchronization in an IEEE-796 Bus Multiprocessor
- FERMTOR Tunable Multiprocessor Architecture
- An Interactive Debugging Environment

OCTOBER 1985

The Engineering Workstation

When does a workstation become an Engineering Support System? Taking a tough user viewpoint, this issue will explore the features and innovations needed to make workstations fully productive. Topics will include: real-time computing, networking and distributed processing, data security, software environments, and the user interface.

DECEMBER 1985

32-Bit MPUs

Based on new product announcements, 1985 is shaping up to be the year of the 32-bit microprocessor. **IEEE Micro** examines this phenomenon in depth with contributions from major international manufacturers, including Motorola, Intel, Zilog, National Semiconductor, and NEC. Other features will include a look at 32-bit buses.

Are there topics of particular interest that you would like to see in **IEEE Micro?** Would you like to submit an article or organize a special issue? Fill out the Reader Interest Card at the back of the magazine.